EECE 5639 Computer Vision I

Lecture 20

- **Circulant tracker**
- **Dynamics-based Tracking**
- **Project 4 is out. Due April 12.**

Next Class

- **SFM**

Tracking by Detection Using Dense Sampling

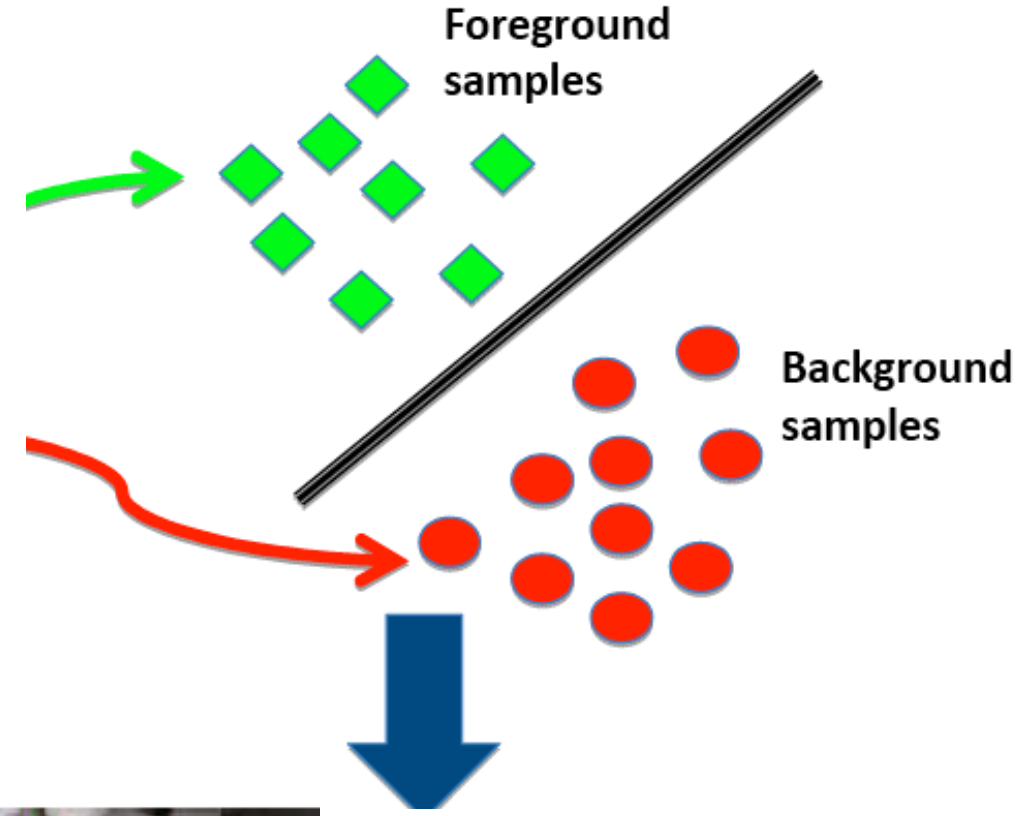
(Henriques, Caseiro, Martins & Batista, '12)

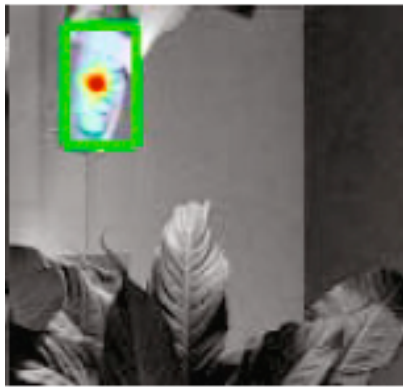
At each frame, collect a set of dense samples around the estimated location of the target.

Samples are continuously labeled (between 1 and 0) using a Gaussian centered at the target location with stdev s

Uses a Kernel Regularized Least Square Classifier

Overview





Classifier Design Using Labeled Data

Given labeled training samples:

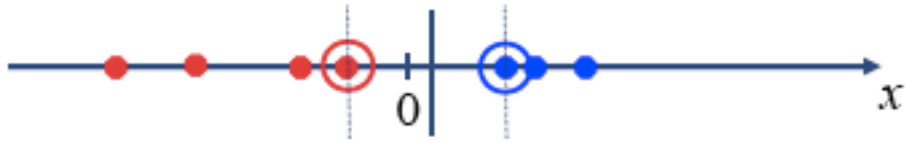
$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)$$

$$y_i \in \{-1, 1\}$$

- Would like to find a classifier function F(x; c) that given a sample x, would return its label.

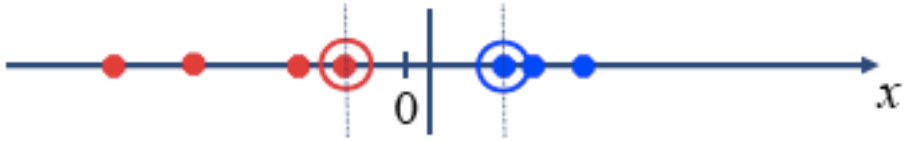
Classifier Using Labeled Data

- Datasets that are linearly separable work out great:



Classifier Using Labeled Data

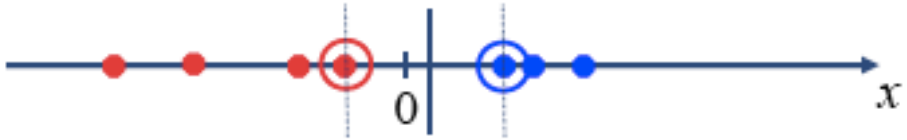
- Datasets that are linearly separable work out great:



- But what if the dataset is just too hard?

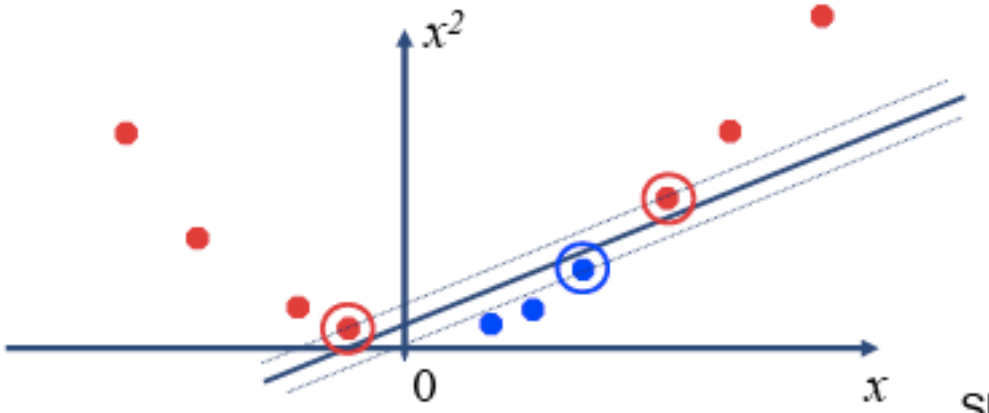
Classifier Using Labeled Data

- Datasets that are linearly separable work out great:



- But what if the dataset is just too hard?

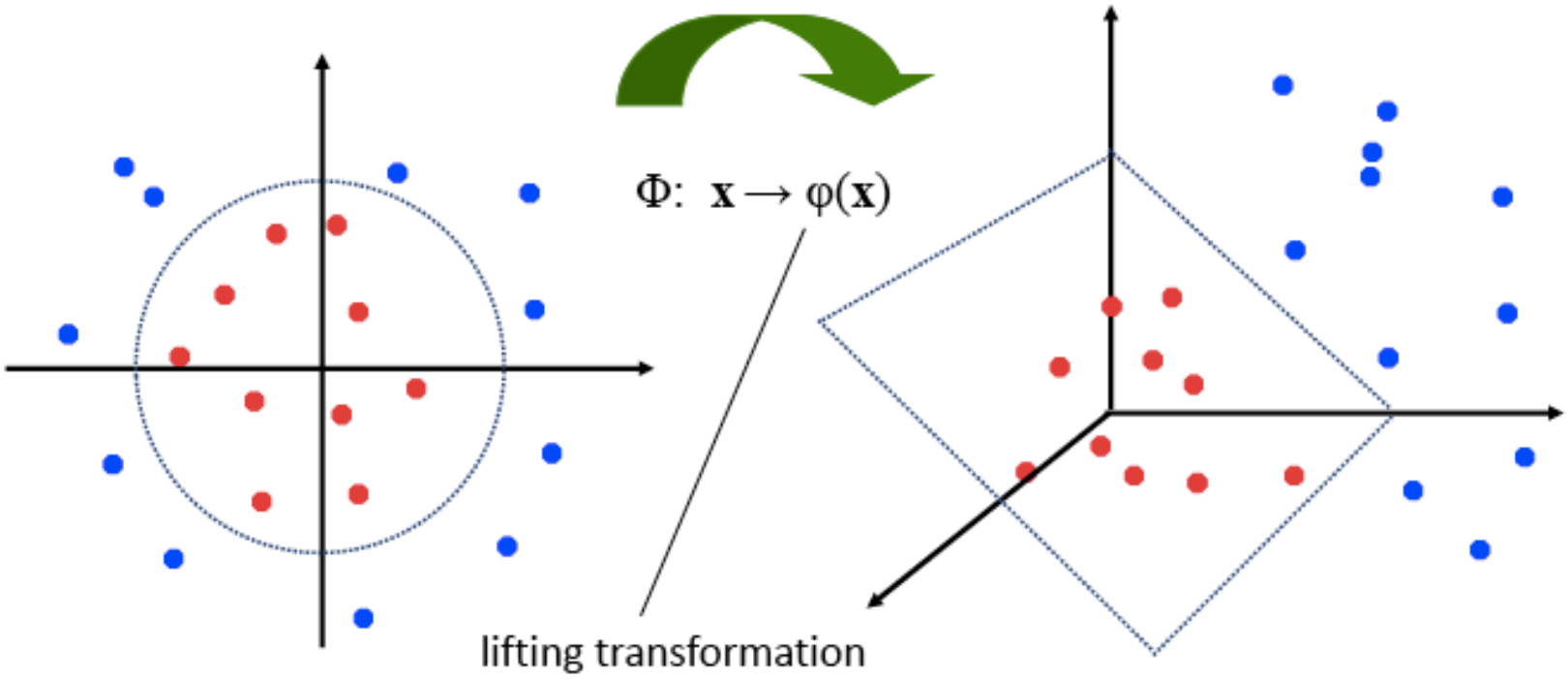
- We can map it to a higher-dimensional space:



Slide credit: Andrew Moore

"Kernel Trick"

- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:



Slide credit: Andrew Moore

A Classifier: Kernel Regularized Least Square Classifier

Given labeled training samples:

$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)$$

- and a "kernel" $\kappa(\mathbf{x}, \mathbf{x}') = \langle \varphi(\mathbf{x}), \varphi(\mathbf{x}') \rangle$

(Non-linear) transformation

dot product

- Would like to find a classifier function f(z; α) that given a sample z, would return its label:

$$y = f(\mathbf{z}; \alpha) = \sum_{i=1}^{l} K(\mathbf{z}, \mathbf{x_i}) \alpha_i$$

the "distances" between z and all the labeled samples

A Classifier: Kernel Regularized Least Square Classifier

- Would like to find a classifier function f(x; α) that given a sample x, would return its label:

$$y = f(\mathbf{z}; \alpha) = \sum_{i=1}^{l} K(\mathbf{z}, \mathbf{x_i})\alpha_i$$

- Pose it as a LSE problem:

$$\min F(\alpha) = \min_{\alpha \in R^l} \frac{1}{2l} \sum_{i=1}^{l} (\mathbf{y} - K\alpha)^T (\mathbf{y} - K\alpha) + \frac{\lambda}{2} \alpha^T K \alpha$$

"good oracle" term

"regularization term" puts a bound on α

A Classifier: Kernel Regularized Least Square Classifier

$$\min F(\alpha) = \min_{\alpha \in R^l} \frac{1}{2l} \sum_{i=1}^{l} (\mathbf{y} - K\alpha)^T (\mathbf{y} - K\alpha) + \frac{\lambda}{2} \alpha^T K \alpha$$

$$\nabla_\alpha F(\alpha) = 0 = \frac{1}{l} \sum_{i=1}^{l} (-K\mathbf{y} + K^2\alpha) + \lambda K\alpha = -K\mathbf{y} + K^2\alpha + \lambda K\alpha$$

$$\boxed{\mathbf{y} = (K + \lambda I)\alpha}$$

Can solve for α☺

Need to invert a LARGE lx $l$matrix ☹

KRLSC

Learning:

Given a set of labeled samples, need to compute α

$$\mathbf{y} = (K + \lambda I)\alpha$$

Matrix Inversion

Testing:

Given an unlabeled sample z, need to compute y:

$$y = f(\mathbf{z}; \alpha) = \sum_{i=1}^{l} K(\mathbf{z}, \mathbf{x_i})\alpha_i = K(\mathbf{z}, \mathbf{x}\cdot)^T \alpha$$

Distances computation, Cross-correlation

Efficiency considerations

We want:

- to use a large number of samples (dense sampling)
- the samples themselves are high dimensional (hundreds, thousands of pixels)
- to learn a new classifier at each frame
- test many unlabeled samples to find a “good” positive example

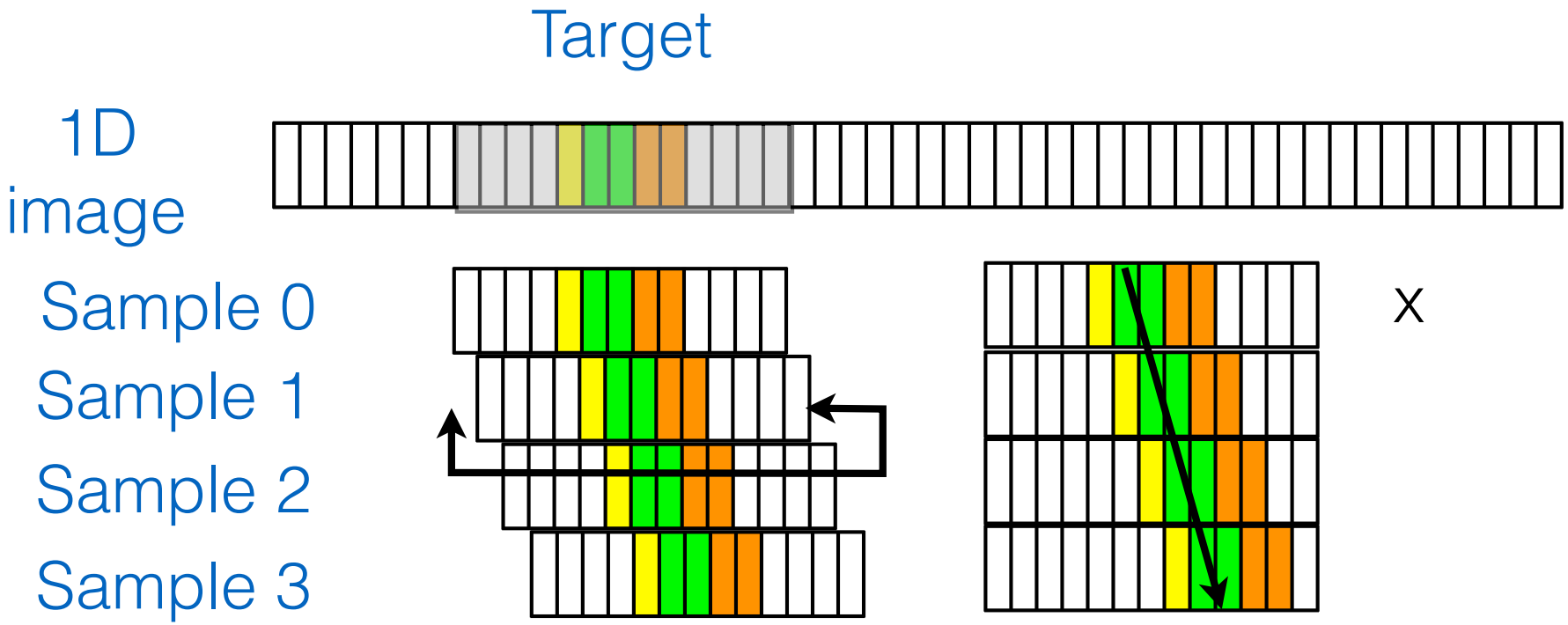
Solution:

- Use “circulant” matrices
- Make computations in the FREQUENCY domain
  - instead of multiplying matrices, we can multiply element by element
  - instead of inverting matrices, we can divide element by element

Price we pay:

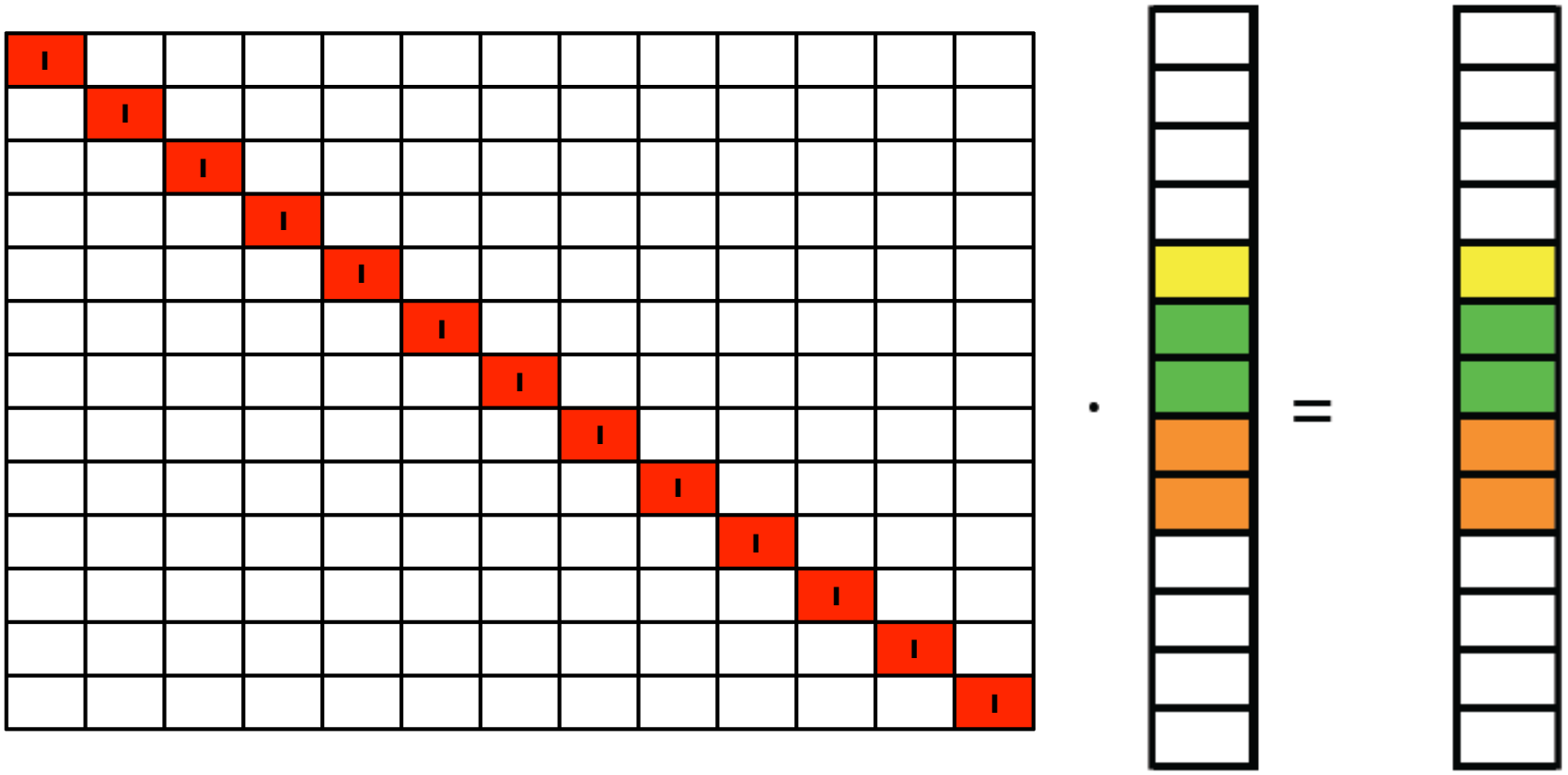
- We need to go to the Frequency domain and come back

“Dense Sampling” = Circulant Matrix



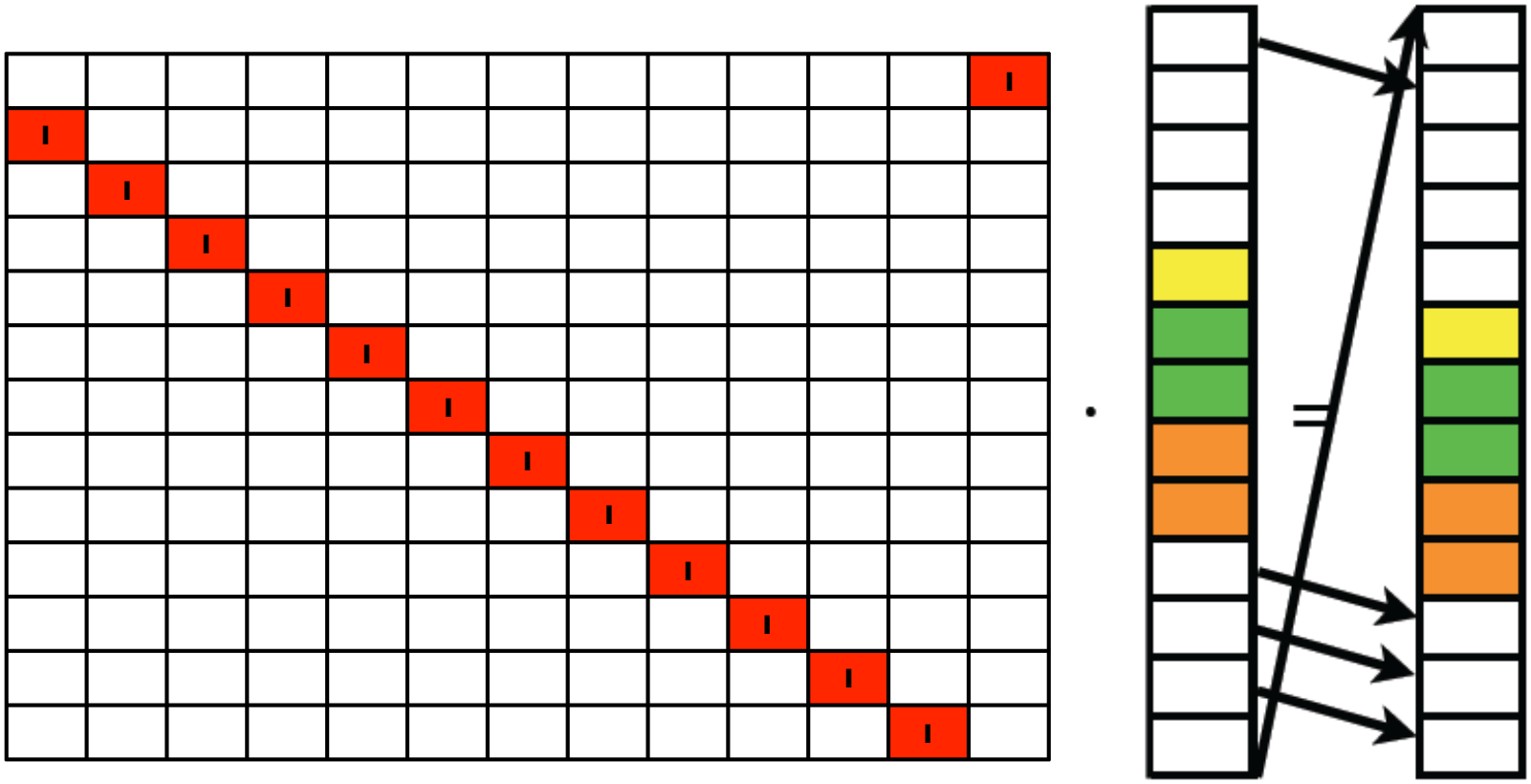
C(x) Circulant Matrix with constant diagonals

$$\mathbf{x_o} = I\mathbf{x} = P^0\mathbf{x_o}$$



Permutation Matrix: P

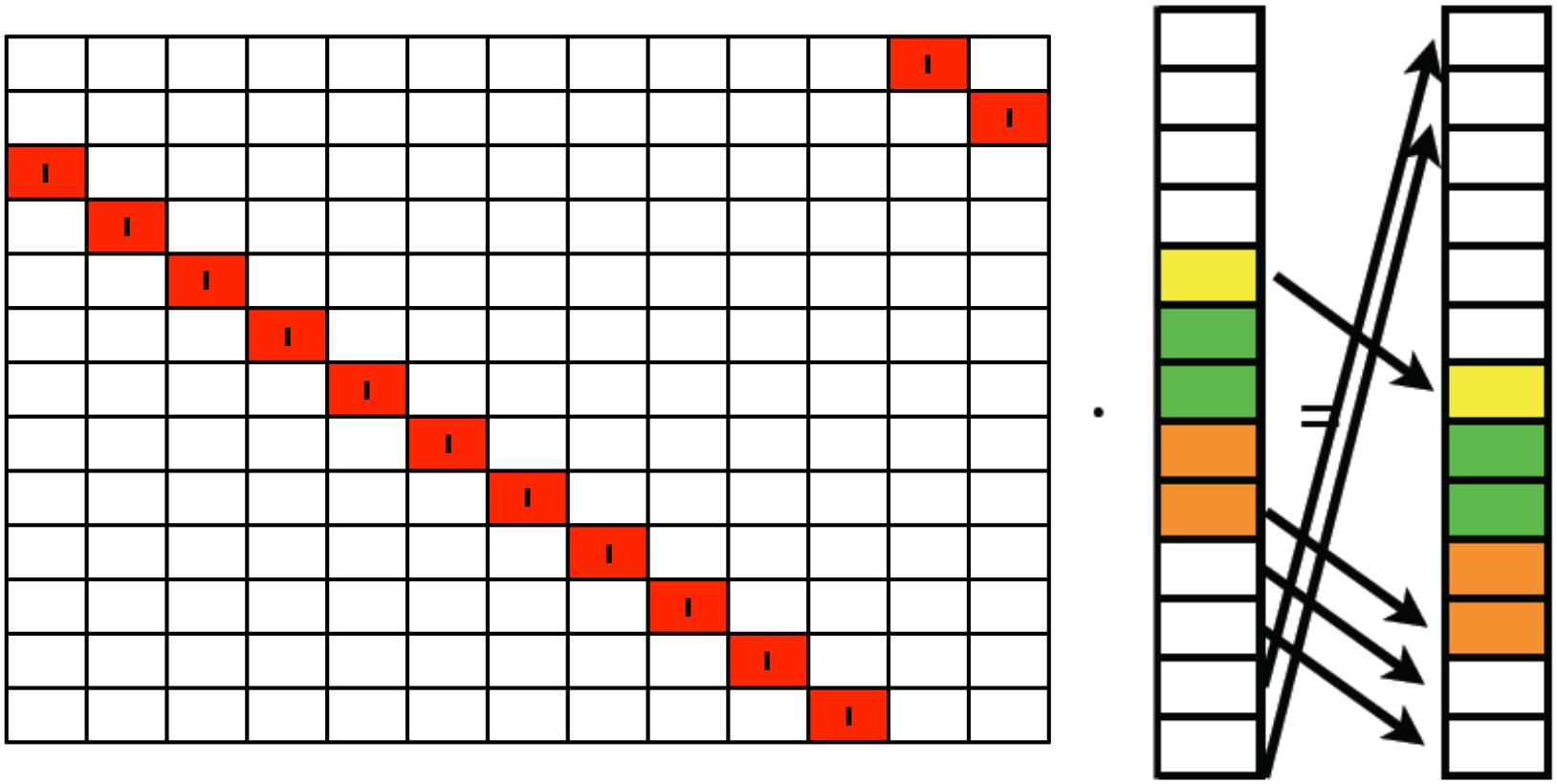
$$\mathbf{x_1} = P\mathbf{x}$$



Only one 1 in each row and in each column

Permutation Matrix: P

$$\mathbf{x_2} = P\mathbf{x_1} = PP\mathbf{x} = P^2\mathbf{x}$$



Only one 1 in each row and in each column

The Frequency Domain

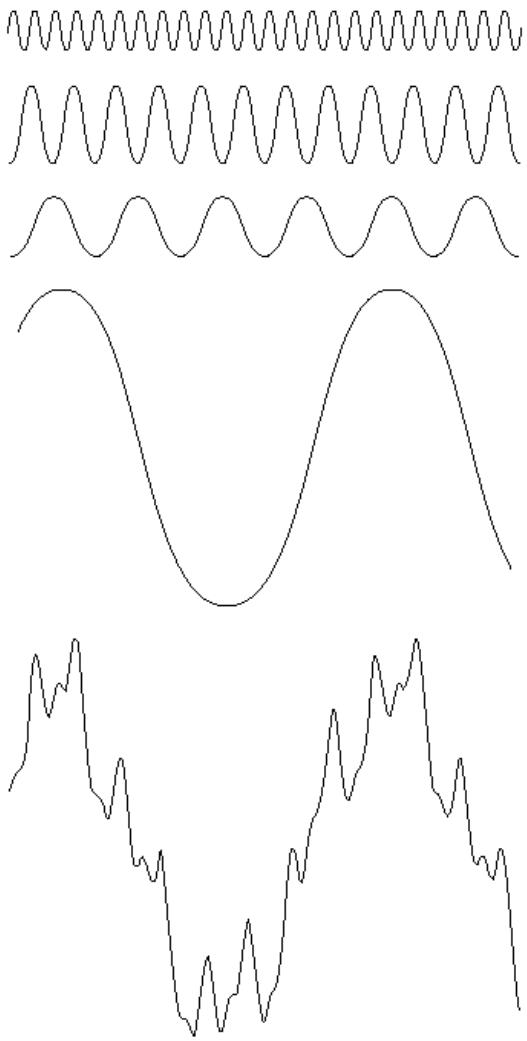


FIGURE 4.1 The function at the bottom is the sum of the four functions above it. Fourier's idea in 1807 that periodic functions could be represented as a weighted sum of sines and cosines was met with skepticism.

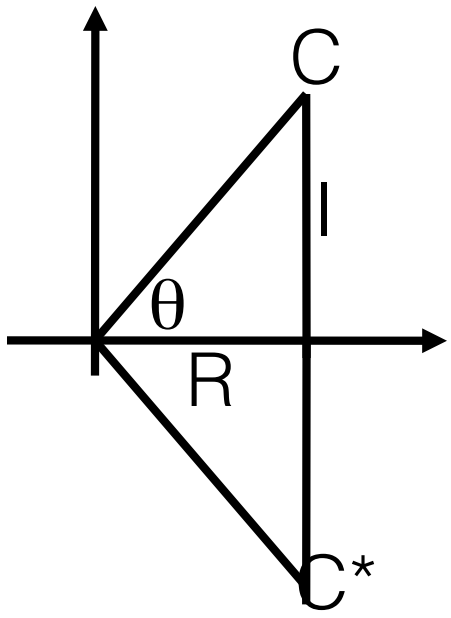
The sum of the four top functions (single frequencies) gives this "noisy" looking signal.

Review: Complex Numbers

Complex Numbers: $C = R + jI$

Conjugate: $C^* = R - jI$

Polar Coordinates: $C = |C|(\cos\theta + j\sin\theta)$



with $|C| = \sqrt{R^2 + I^2}$

$$\tan\theta = \frac{I}{R}$$

Using Euler's formula: $C = |C|e^{j\theta}$

$$e^{j\theta} = \cos\theta + j\sin\theta$$

Sampling f(x)

Samples of f(x) and F(u) are not necessarily at integer values, but they should be at equally spaced points:

$$f(x) \doteq f(x_o + k\Delta x)$$ With **k** integer

$$F(u) \doteq F(u\Delta u)$$ With **u** integer

Δx and **Δu** are related: $$\Delta u = \frac{1}{M\Delta x}$$

Discrete Fourier Transform (DFT)

One Dimension :

Let f(x) be a discrete function of x = 0,1,2,…,M-1

$$F(u) = \sum_{x=0}^{x=M-1} f(x) e^{-j2\pi ux/M} \quad u = 0, 1, \ldots, M-1$$

$$f(x) = \frac{1}{M} \sum_{u=0}^{u=M-1} F(u) e^{+j2\pi ux/M} \quad x = 0, 1, \ldots, M-1$$

NOTE: f(x) is real BUT F(u) will be, in general, complex

Euler's Formula and the Frequency Domain

$$F(u) = \sum_{x=0}^{x=M-1} f(x) e^{-j2\pi ux/M} \quad u = 0, 1, \ldots, M-1$$

Euler's formula: $e^{j\theta} = \cos\theta + j \sin\theta$

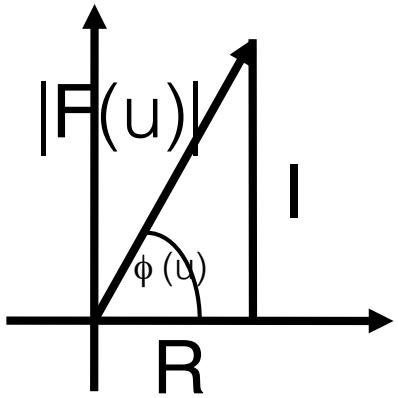
$$F(u) = \sum_{x=0}^{x=M-1} f(x)[\cos\frac{2\pi ux}{M} - j \sin\frac{2\pi ux}{M}] \quad u = 0, 1, \ldots, M-1$$

- For each value of u, we need to sum ALL values of f(x)
- u determines the frequency of the transform

Fourier Transform

$$F(u) = R(u) + jI(u)$$

$$F(u) = |F(u)|e^{j\phi(u)}$$



where $|F(u)| = \sqrt{R^2(u) + I^2(u)}$ "spectrum of f(x)"

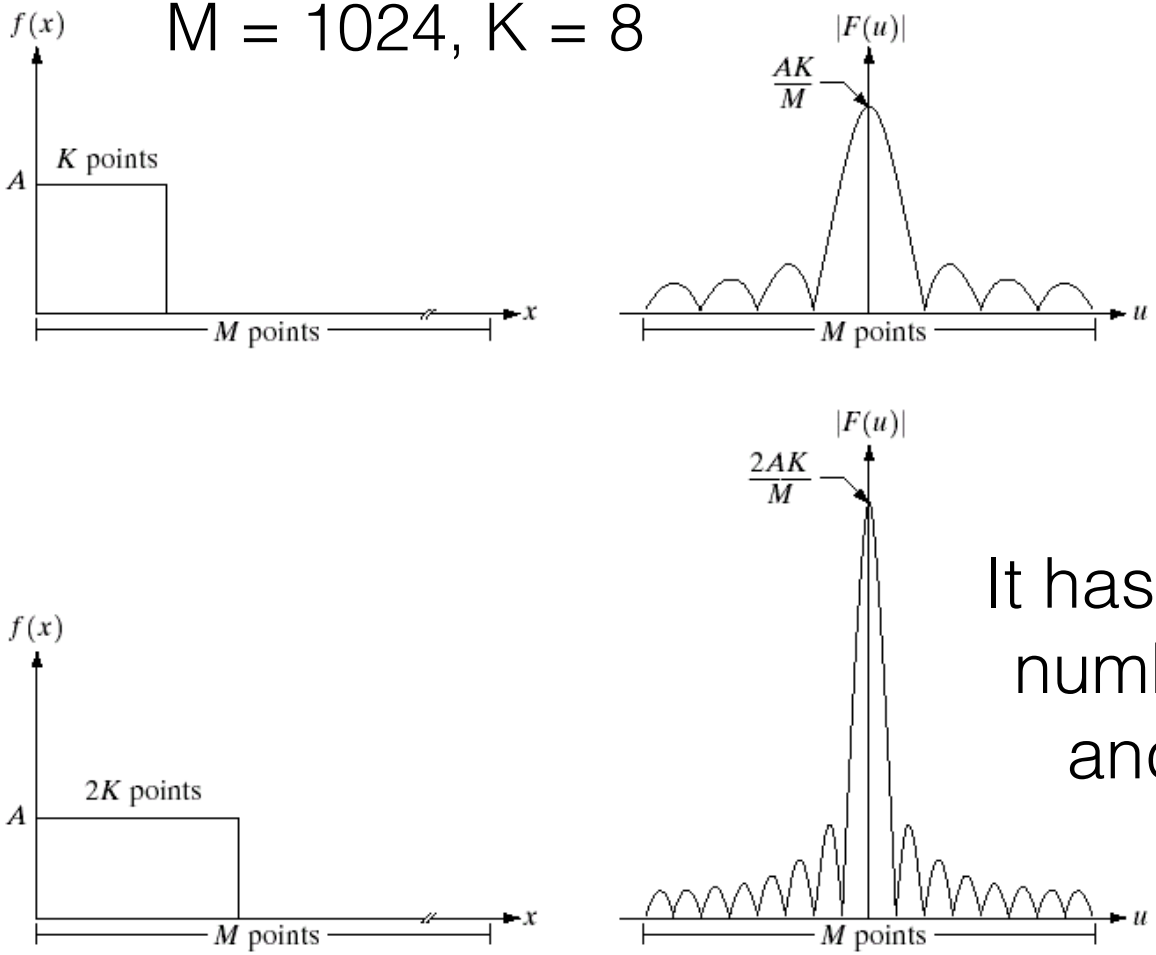
$P(u) = |F(u)|^2$ "power spectrum or spectral density of f(x)"

$\phi(u) = \tan^{-1}\frac{I(u)}{R(u)}$ "phase angle"

1D DFT Example

The functions are **discrete**, only drawn as continuous for visualization!

M = 1024, K = 8



a b
c d

FIGURE 4.2 (a) A discrete function of M points, and (b) its Fourier spectrum. (c) A discrete function with twice the number of nonzero points, and (d) its Fourier spectrum.

It has doubled the number of zeros and doubled height

FT and Convolution

$$\mathcal{F}[f(t) * h(t)] = F(u)H(u)$$
$$\mathcal{F}[f(t)h(t)] = F(u) * H(u)$$

$$\boxed{f(t) * h(t) = \mathcal{F}^{-1}\left[\mathcal{F}(u).\mathcal{H}(u)\right]}$$

Kernels & Circulant Matrices

$$C(\mathbf{u}) = \begin{bmatrix} u_0 & u_1 & u_2 & \cdots & u_{n-1} \\ u_{n-1} & u_0 & u_1 & \cdots & u_{n-2} \\ u_{n-2} & u_{n-1} & u_0 & \cdots & u_{n-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ u_1 & u_2 & u_3 & \cdots & u_0 \end{bmatrix}.$$

$$c_{ij} = u_{(j-i) \bmod n}$$

n is the number of samples

Kernels & Circulant Matrices

Unitary Matrix: $U^H = (U^*)^T = U^{-1}$

κ is unitarily invariant if $\kappa(\mathbf{x}, \mathbf{x}') = \kappa(U\mathbf{x}, U\mathbf{x}')$ for any unitary matrix U

Permutations are unitary matrices.

Kernels & Circulant Matrices

$$\mathbf{x}_i = P^i \mathbf{x}, \quad \forall i = 0, \ldots, n-1$$

Theorem 1. *The matrix K with elements $K_{ij} = \kappa(P^i\mathbf{x}, P^j\mathbf{x})$ is circulant if κ is a unitarily invariant kernel.*

Proof: $K_{ij} = \kappa(P^i\mathbf{x}, P^j\mathbf{x})$

κ is unitarily invariant if $\kappa(\mathbf{x}, \mathbf{x}') = \kappa(U\mathbf{x}, U\mathbf{x}')$ for any unitary matrix U

Permutations are unitary matrices.

$$K_{ij} = \kappa(P^{-i}P^i\mathbf{x}, P^{-i}P^j\mathbf{x}) = \kappa(\mathbf{x}, P^{j-i}\mathbf{x})$$

depends only on (j-i)mod n.

QED

Circulant Matrices and FFT

$$C(\mathbf{u}) = \begin{bmatrix} u_0 & u_1 & u_2 & \cdots & u_{n-1} \\ u_{n-1} & u_0 & u_1 & \cdots & u_{n-2} \\ u_{n-2} & u_{n-1} & u_0 & \cdots & u_{n-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ u_1 & u_2 & u_3 & \cdots & u_0 \end{bmatrix}. \qquad v = \begin{bmatrix} v_o \\ v_2 \\ \vdots \\ v_{n-1} \end{bmatrix}$$

$$C(u).v = u * v \quad \text{Circulant convolution}$$

$$\mathcal{F}\left(C(u).v\right) = \mathcal{F}^*(u) \odot \mathcal{F}(v) \quad \text{element-wise product}$$

$$v = \mathcal{F}^{-1}\left[\frac{\mathcal{F}\left(C(u).v\right)}{\mathcal{F}^*(u)}\right] \quad \text{element-wise division}$$

Circulant Matrices, Kernels + FFT = Efficient!

Learning:

Given a set of labeled samples, need to compute α

$$\boxed{\mathbf{y} = (K + \lambda I)\alpha}$$

Matrix Inversion

$K = C(\mathbf{k})$ Circulant matrix with columns:

$$k_i = \kappa(\mathbf{x}, P^i\mathbf{x}), \quad \forall i = 0, \ldots, n-1$$

$$\boldsymbol{\alpha} = \mathcal{F}^{-1}\left(\frac{\mathcal{F}(\mathbf{y})}{\mathcal{F}(\mathbf{k}) + \lambda}\right), \text{element-wise division!}$$

For $n \times n$ images, the proposed algorithm has a complexity of only $\mathcal{O}(n^2 \log n)$

Circulant Matrices, Kernels + FFT = Efficient!

- Testing:
  - Given an unlabeled sample z, need to compute y:

$$y = f(\mathbf{z}; \alpha) = \sum_{i=1}^{l} K(\mathbf{z}, \mathbf{x_i})\alpha_i = K(\mathbf{z}, \mathbf{x_i})^T \alpha$$

Distances computation, Cross-correlation

$$K = C(\mathbf{k})$$

$$k_i = \kappa(\mathbf{x}, P^i\mathbf{x}), \quad \forall i = 0, \ldots, n-1$$

$$\bar{k}_i = \kappa(\mathbf{z}, P^i\mathbf{x})$$

$$\hat{\mathbf{y}} = \mathcal{F}^{-1}\left(\mathcal{F}(\bar{\mathbf{k}}) \odot \mathcal{F}(\boldsymbol{\alpha})\right)$$

element-wise multiplication!

For $n \times n$ images, the proposed algorithm has a complexity of only $\mathcal{O}(n^2 \log n)$

Fast Kernel Computation

Linear case: no Kernel Trick, just dot product

$$\kappa(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle$$

$$\mathbf{w} = \mathcal{F}^{-1}\left(\frac{\mathcal{F}(\mathbf{x}) \odot \mathcal{F}^*(\mathbf{y})}{\mathcal{F}(\mathbf{x}) \odot \mathcal{F}^*(\mathbf{x}) + \lambda}\right)$$

Fast Kernel Computation

Gaussian Kernel

$$\mathbf{k}^{\text{gauss}} = \exp\left(-\frac{1}{\sigma^2}\left(\|\mathbf{x}\|^2 + \|\mathbf{x}'\|^2 - 2\mathcal{F}^{-1}\left(\mathcal{F}(\mathbf{x}) \odot \mathcal{F}^*(\mathbf{x}')\right)\right)\right)$$

Implementation Details

Images are not really periodic ... bound them using a sinusoidal window:

$$\left(x_{ij}^{\text{raw}} - 0.5\right) \sin(\pi i/n) \sin(\pi j/n), \quad \forall i, j = 0, \ldots, n-1$$

- Use a “continuous” class label to yield smooth spatial responses:

$$y_{ij} = \exp\left(-\left((i-i')^2 + (j-j')^2\right)/s^2\right), \quad \forall i, j = 0, \ldots, n-1$$

- “Incorporates temporal memory” by linearly interpolating the new parameters with the ones from the previous frame:

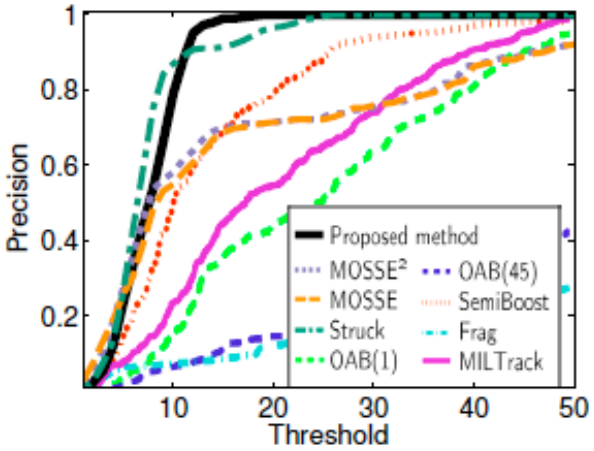
```
alphaf = (1 - interp_factor) * alphaf + interp_factor * new_alphaf;
       z = (1 - interp_factor) * z + interp_factor * new_z;
```

CODE

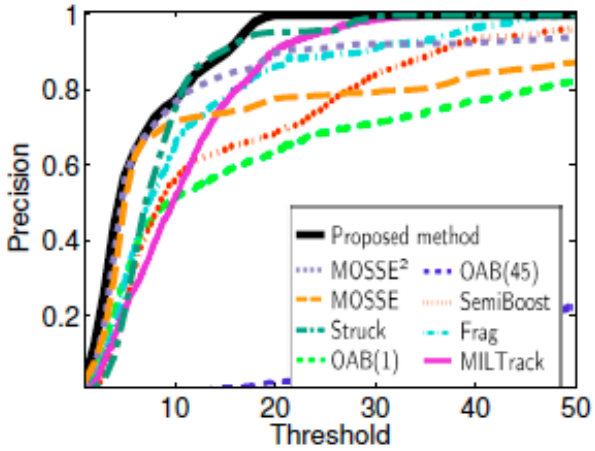
Algorithm 1 . MATLAB code for our tracker, using a Gaussian kernel
It is possible to reuse some values, reducing the number of FFT calls. An implementation with GUI is available at: `http://www.isr.uc.pt/~henriques/`

```
% Training image x (current frame) and test image z (next frame)
% must be pre-processed with a cosine window. y has a Gaussian
% shape centered on the target. x, y and z are M-by-N matrices.
% All FFT operations are standard in MATLAB.

function alphaf = training(x, y, sigma, lambda)        % Eq. 7
  k = dgk(x, x, sigma);
  alphaf = fft2(y) ./ (fft2(k) + lambda);
end

function responses = detection(alphaf, x, z, sigma)   % Eq. 9
  k = dgk(z, x, sigma);
  responses = real(ifft2(alphaf .* fft2(k)));
end

function k = dgk(x1, x2, sigma)                        % Eq. 16
  c = fftshift(ifft2(fft2(x1) .* conj(fft2(x2))));
  d = x1(:)'*x1(:) + x2(:)'*x2(:) - 2*c;
  k = exp(-1 / sigma^2 * abs(d) / numel(x1));
end
```

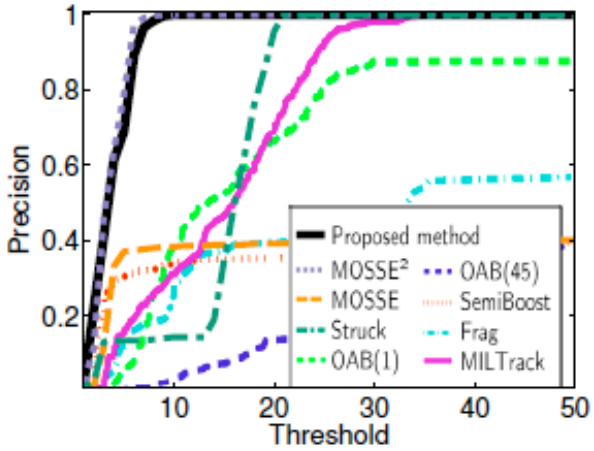
Performance (300 fps)



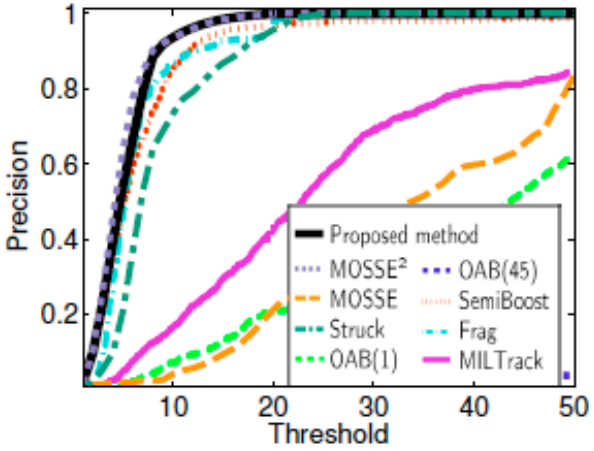
(a) coke11



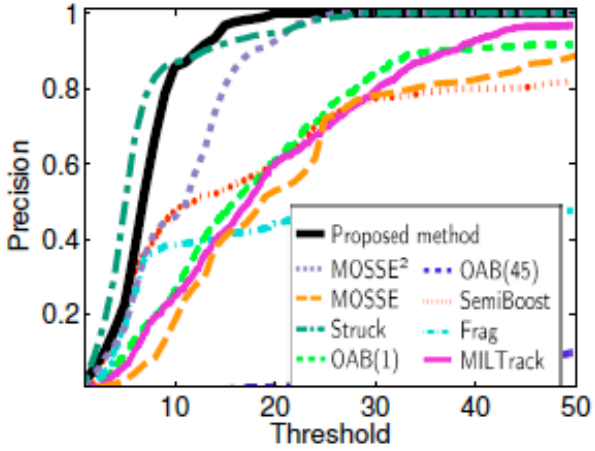
(b) sylvester



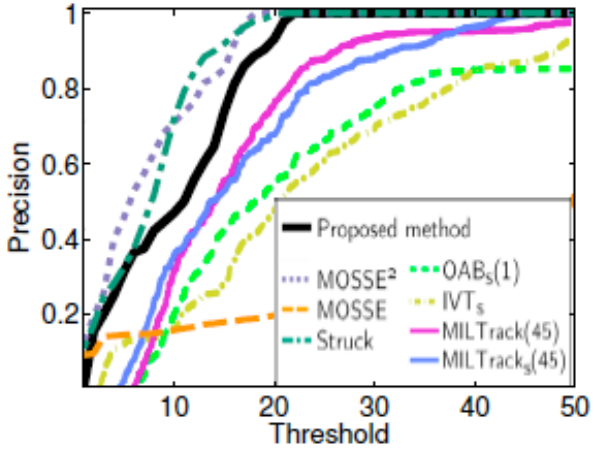
(c) dollar



(d) faceocc

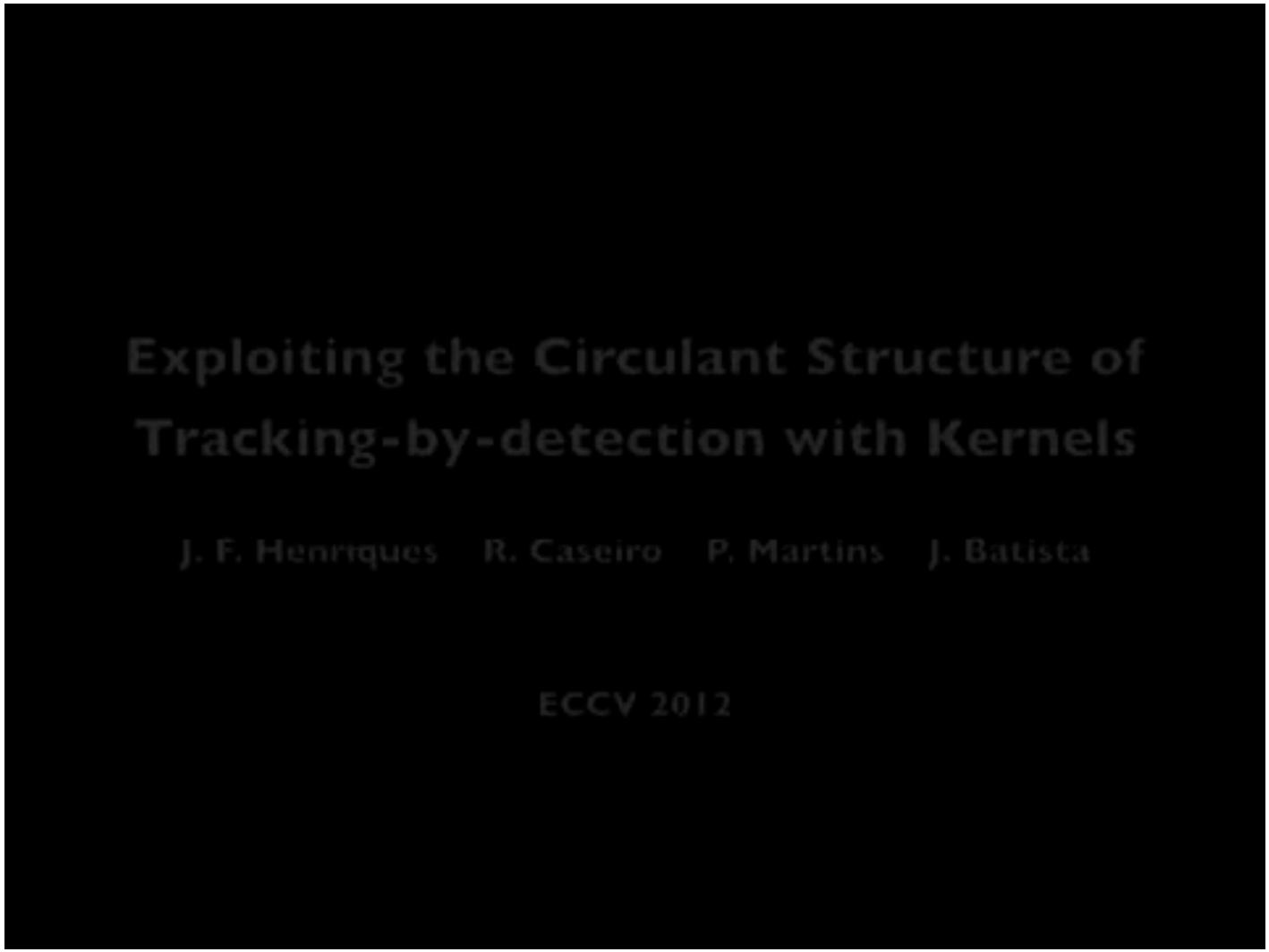


(e) faceocc2



(f) twinings

Videos

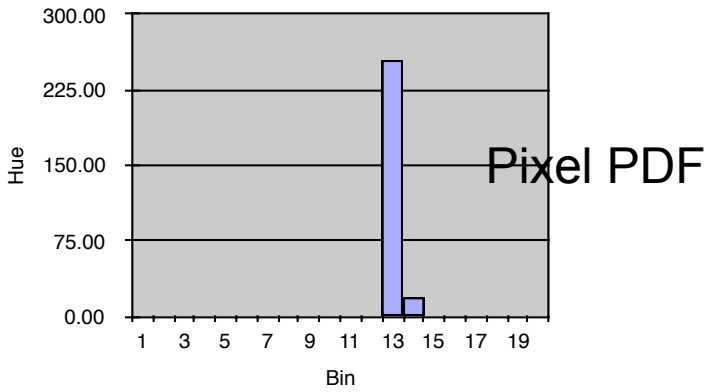


http://www.youtube.com/watch?v=sZXKnPUhAi8

Tracking Using Tokens

Color-based tracking

1.Set initial search window

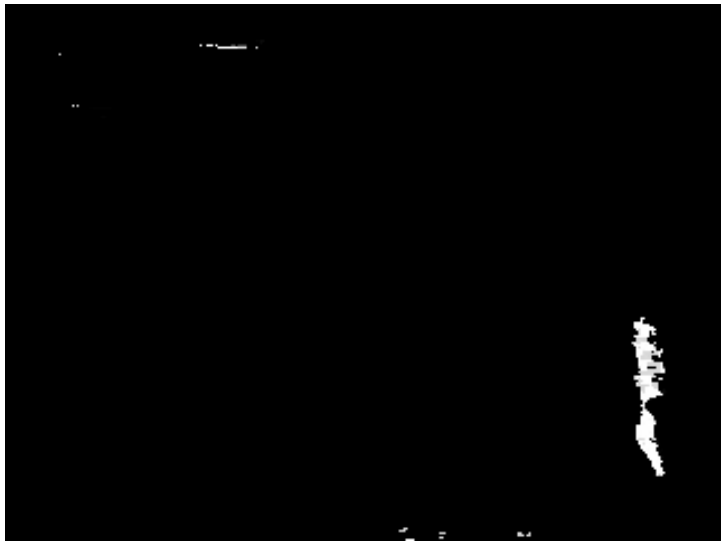


2.Calculate histogram

3.Original image

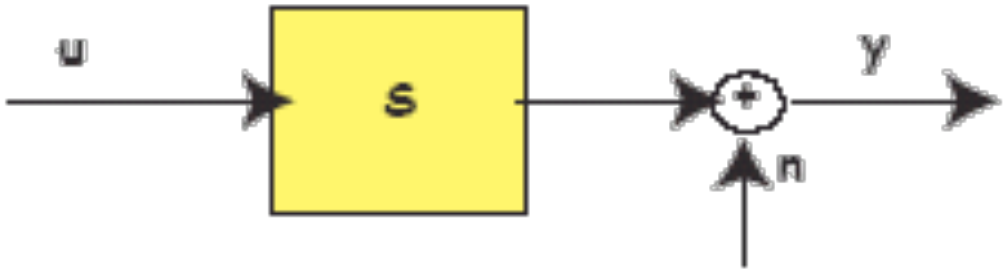
4.Probability image

Occlusion Fragility



Dynamical System

*A **dynamical system** is a mathematical formalization for any fixed "rule" which describes the time dependence of a point's position in its ambient space.*



- A dynamical system has a state determined by a collection of numbers.
- The evolution rule describes what future states follow the current state.

Multiframe Tracking

Use simple *dynamical models* to integrate information over time and make predictions with

- Kalman, Extended Kalman, Unscented Kalman Filters
- Particle, Unscented Particle Filters
- Hankel matrix

State: position, velocity, acceleration
Rule: constant acceleration

References

"Computer Vision, A Modern Approach" by Forsyth and Ponce (Chapter 17)

"Three Dimensional Computer Vision", by O. Faugeras

"Applied Optimal Estimation", ed. by A. Gelb

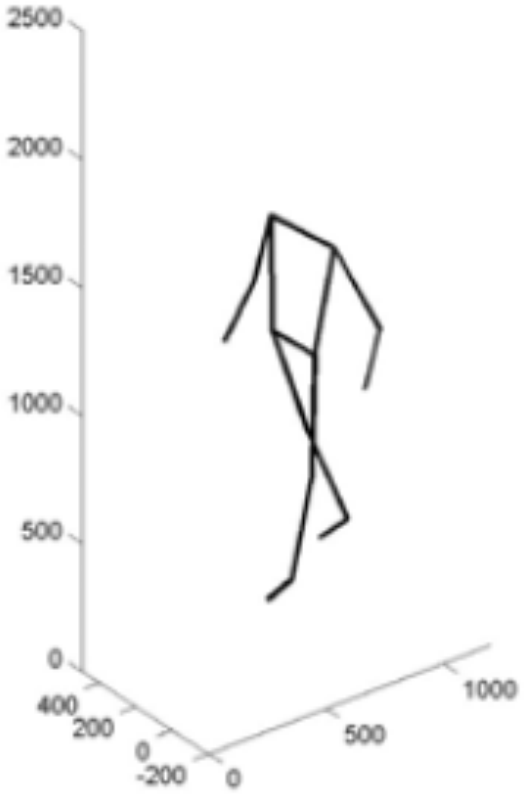
Tracking

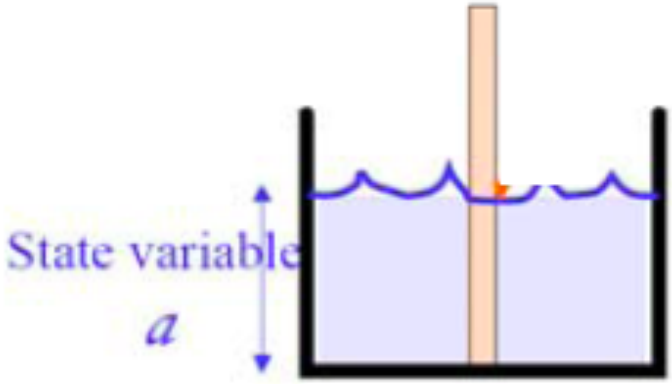
(adapted from D. Forsyth)

Very general model:

- Moving objects have an underlying “state” x
- There are noisy measurements y
- The measurements are functions of the current state
- There is a clock
  - at each tick, the state changes
  - at each tick, we get a new observation

State vs. Observation

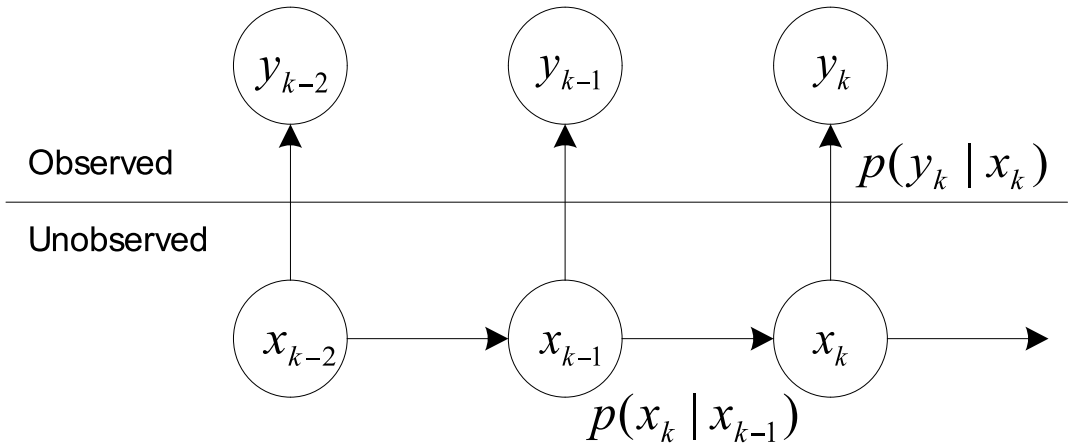




State (hidden): parameters of interest
Measurement: what we get to directly observe

Dynamic State Space Model

The state x_k, the observation y_k at time k



- Assumptions
  - The current state depends only on n past states.

$$p(x_k|x_{k-1}, \ldots, x_{-\infty}) = p(x_k|x_{k-1}, \ldots, x_{k-n})$$

  - The observations are independent, given states

$$p(y_i, y_j, \ldots, y_k|x_i) = p(y_i|x_i)p(y_j, \ldots, y_k|x_i)$$

Tracking

(adapted from D. Forsyth)

Examples:

object is ball:

state is 3D position+velocity,

measurements are stereo pairs

object is person:

state is body configuration,

measurements are frames,

clock is in camera (30 fps)

Filtering & Prediction

Filtering:

Estimation of the “state vector” at the current time.

Prediction:

Estimation of the “state vector” at a future time.

Filtering and Prediction are closely related!

What makes a GOOD estimate of **x** ?

A good estimate: $\widehat{x}$

Should be UNBIASED: $E(\widehat{x}) = E(x)$

Minimum Variance: $\widehat{x} = \arg\min_{x_e} \mathrm{Var}(x_e)$

Consistent: $\widehat{x} \longrightarrow x$

As the number of measurements goes to infinity

PROBLEM:

Let:

x be the state vector (nx1)

y be the measurements (lx1)

ν be the measurement error (lx1)

such that:

$$y = Mx + \nu$$

Find an *unbiased, min. variance, consistent* estimate of x based on the measurements y.

Least Squares Approach (assuming l > n)

$$\min_{\hat{x}} J = \min_{\hat{x}} (y - M\hat{x})^T (y - M\hat{x})$$

$$J = y^T y - y^T M\hat{x} - \hat{x}^T M^T y + \hat{x}^T M^T M\hat{x}$$

Nec. Condition:

$$\frac{\partial J}{\partial \hat{x}} = 0$$

$$\hat{x} = (M^T M)^{-1} M^T y$$

Weighted Least Squares Approach (assuming l > n)

$$\min_{\hat{x}} J = \min_{\hat{x}} (y - M\hat{x})^T R^{-1} (y - M\hat{x})$$

Condition:

$$\frac{\partial J}{\partial \hat{x}} = 0$$

$$\hat{x} = (M^T R^{-1} M)^{-1} M^T R^{-1} y$$

Maximum Likelihood Approach

$$\hat{x} \text{ such that } \max p(y|x)$$

Assuming: $\nu \sim N(0, \Sigma_m)$

$$p(y|x) = \frac{1}{(2\pi)^{l/2}|\Sigma_m|^{1}/2} \exp\{-1/2(y-Mx)^T \Sigma_m^{-1}(y-Mx)\}$$

$$\max p(y|x) \Leftrightarrow \min(y - Mx)\Sigma_m^{-1}(y - Mx)$$

WLS with $\mathbf{R} = \Sigma_m$!!

Bayesian Approach

$$\widehat{x} \text{ such that } \max p(x|y)$$

Bayes:

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}$$

Assuming p(**x**) is UNIFORM:

$$\max p(x|y) \Leftrightarrow \max p(y|x)$$

Min. Variance Bayes Estimate

$$\min_{\hat{x}} J$$

$$J = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \dots \int_{-\infty}^{\infty} (\hat{x}-x)^T S(\hat{x}-x) p(x|y) dx_1 dx_2 \dots dx_n$$

Where S is an arbitrary matrix S > 0

Nec. Condition:

$$\frac{\partial J}{\partial \hat{x}} = 0$$

Min. Variance Bayes Estimate

$$J = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\dots\int_{-\infty}^{\infty}(\hat{x}-x)^T S(\hat{x}-x)p(x|y)dx_1dx_2\dots dx_n$$

$$\frac{\partial J}{\partial \hat{x}} = 0$$

$$\hat{x}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\dots\int_{-\infty}^{\infty}p(x|y)dx_1dx_2\dots dx_n =$$

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\dots\int_{-\infty}^{\infty}xp(x|y)dx_1dx_2\dots dx_n$$

$$\hat{x} = E(x|y)$$

Min. Variance Bayes Estimate

$$\hat{x} = E(x|y)$$

If $\mathbf{x} \sim N(\mathbf{x}_o, P_o)$ and $\boldsymbol{\nu} \sim N(0, R)$

$$\hat{x} = (P_o^{-1} + M^T R^{-1} M)^{-1} M^T R^{-1} y$$

Recursive Filters

Main Idea:

Update the estimate based on the LAST measurement, instead of recomputing it from the beginning.

Example: Recursive Average

Estimate a CONSTANT x based on k measurements:

$y_i = x + v_i \quad i = 1, \ldots, k$

Non-recursive:

$$\hat{x}_k = \frac{1}{k} \sum_{i=1}^{k} y_i$$

Recursive:

$$\hat{x}_k = \frac{1}{k}[(k-1)\hat{x}_{k-1} + y_k]$$

Example: Recursive Average

Recursive estimate of a CONSTANT x based on k measurements:

$$\hat{x}_k = \frac{1}{k}[(k-1)\hat{x}_{k-1} + y_k]$$

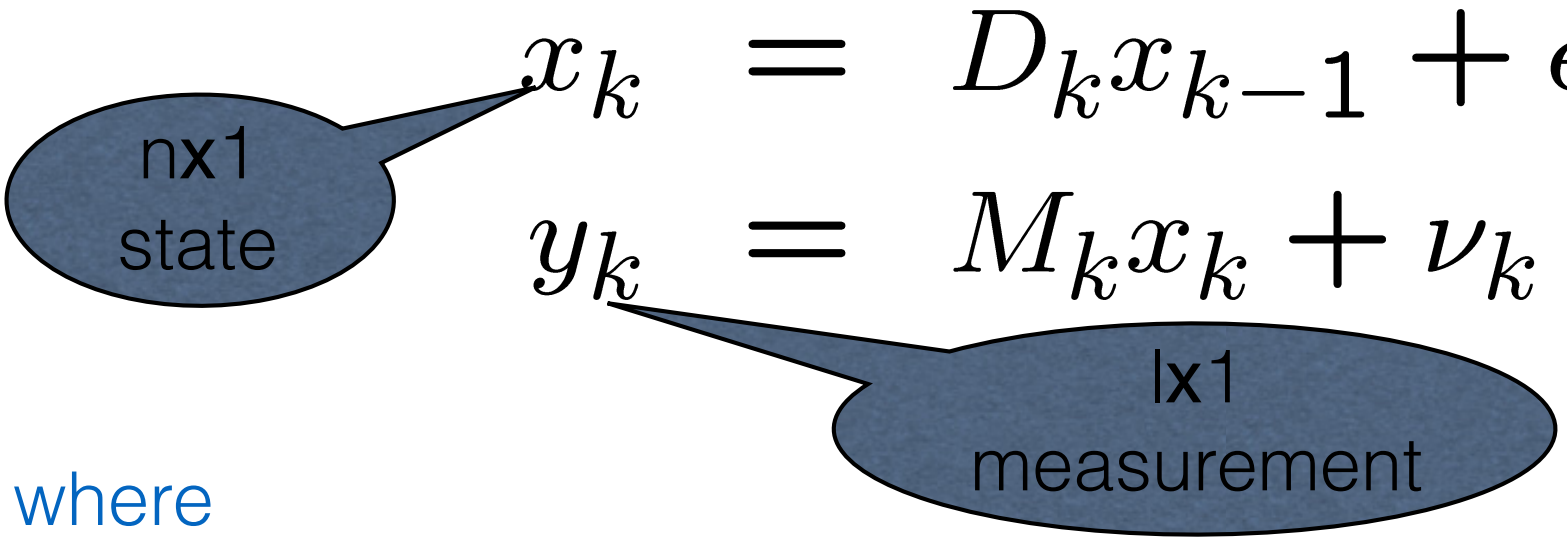
$$\hat{x}_k = \hat{x}_{k-1} + \frac{1}{k}[y_k - \hat{x}_{k-1}]$$

Previous estimate

Measurement Residual

Discrete KALMAN Filter

Consider a LINEAR DYNAMIC system:



$$x_k = D_k x_{k-1} + \epsilon_{k-1}$$

$$y_k = M_k x_k + \nu_k$$

where

$$\epsilon_k \sim N(0, \Sigma_{dk}) \quad \nu_k \sim N(0, \Sigma_{mk})$$

are UNCORRELATED noise

Discrete Kalman Filter

Notation:

Estimate BEFORE measurement k: $\hat{x}_k(-)$

Estimate AFTER measurement k: $\hat{x}_k(+)$

Estimate ERROR BEFORE measurement k:

$$\tilde{x}_k(-) = \hat{x}_k(-) - x_k$$

Estimate ERROR AFTER measurement k:

$$\tilde{x}_k(+) = \hat{x}_k(+) - x_k$$

Discrete Kalman Filter

Want a LINEAR recursive update equation:

$$\hat{x}_k(+) = K'_k \hat{x}_k(-) + K_k y_k$$

Discrete Kalman Filter

We want to find K'_k and K_k so that the estimate below is unbiased and min. variance.

$$\hat{x}_k(+) = K'_k \hat{x}_k(-) + K_k y_k$$

Unbiased ,

$$E[\tilde{x}_k(+)] = E[\tilde{x}_k(-)] = 0$$

Using

$$\tilde{x}_k(-) = \hat{x}_k(-) - x_k$$

$$\tilde{x}_k(+) = \hat{x}_k(+) - x_k$$

$$\hat{x}_k(+) = K'_k \hat{x}_k(-) + K_k y_k$$

$$y_k = M_k x_k + \nu_k$$

$$\tilde{x}_k(+) = K'_k[\tilde{x}_k(-) + x_k] +$$

$$K_k[M_k x_k + \nu_k] - x_k$$

$$\tilde{x}_k(+) = K'_k[\tilde{x}_k(-) + x_k] +$$

$$K_k[M_k x_k + \nu_k] - x_k$$

$$= (K'_k + K_k M_k - I)x_k +$$

$$K'_k \tilde{x}_k(-) + K_k \nu_k$$

$$\tilde{x}_k(+) = (K'_k + K_k M_k - I)x_k + K'_k \tilde{x}_k(-) + K_k \nu_k$$

Imposing that for every $\mathbf{x}_k$ $\quad E[\tilde{x}_k] = 0$

$$(K'_k + K_k M_k - I) = 0$$

$$\boxed{K'_k = I - K_k M_k}$$

Then the estimate is given by:

$$\hat{x}_k(+) = [I - K_k M_k]\hat{x}_k(-) + K_k y_k$$

$$\hat{x}_k(+) = \hat{x}_k(-) + K_k[y_k - M_k\hat{x}_k(-)]$$

Previous estimate

Measurement Residual

And the estimation error is given by:

$$\tilde{x}_k(+) = [I - K_k M_k]\tilde{x}_k(-) + K_k \nu_k$$

Min Error Covariance:

$$\min \text{trace} P_k(+) = \text{trace} E[\tilde{x}_k(+)\tilde{x}_k(+)^T]$$

$$\tilde{x}_k(+) = [I - K_k M_k]\tilde{x}_k(-) + K_k \nu_k$$

$$\tilde{x}_k(+)\tilde{x}_k(+)^T = \{[I-K_k M_k]\tilde{x}_k(-)+K_k\nu_k\}\{\tilde{x}_k(-)^T[I-K_k M_k]^T+\nu_k^T K_k^T\}$$

$$E[\tilde{x}_k \tilde{x}_k^T] = P_k \qquad E[\nu_k {\nu_k}^T] = \Sigma_{mk} \qquad E[\nu_k \tilde{x}_k^T] = 0$$

$$P_k(+) = [I-K_k M_k]P_k(-)[I-K_k M_k]^T+K_k\Sigma_{mk}K_k^T$$

Using that for B symmetric $\dfrac{\partial \text{trace}(ABA^T)}{\partial A} = 2AB$

$$P_k(+) = [I - K_k M_k] P_k(-) [I - K_k M_k]^T + K_k \Sigma_{m_k} K_k^T$$

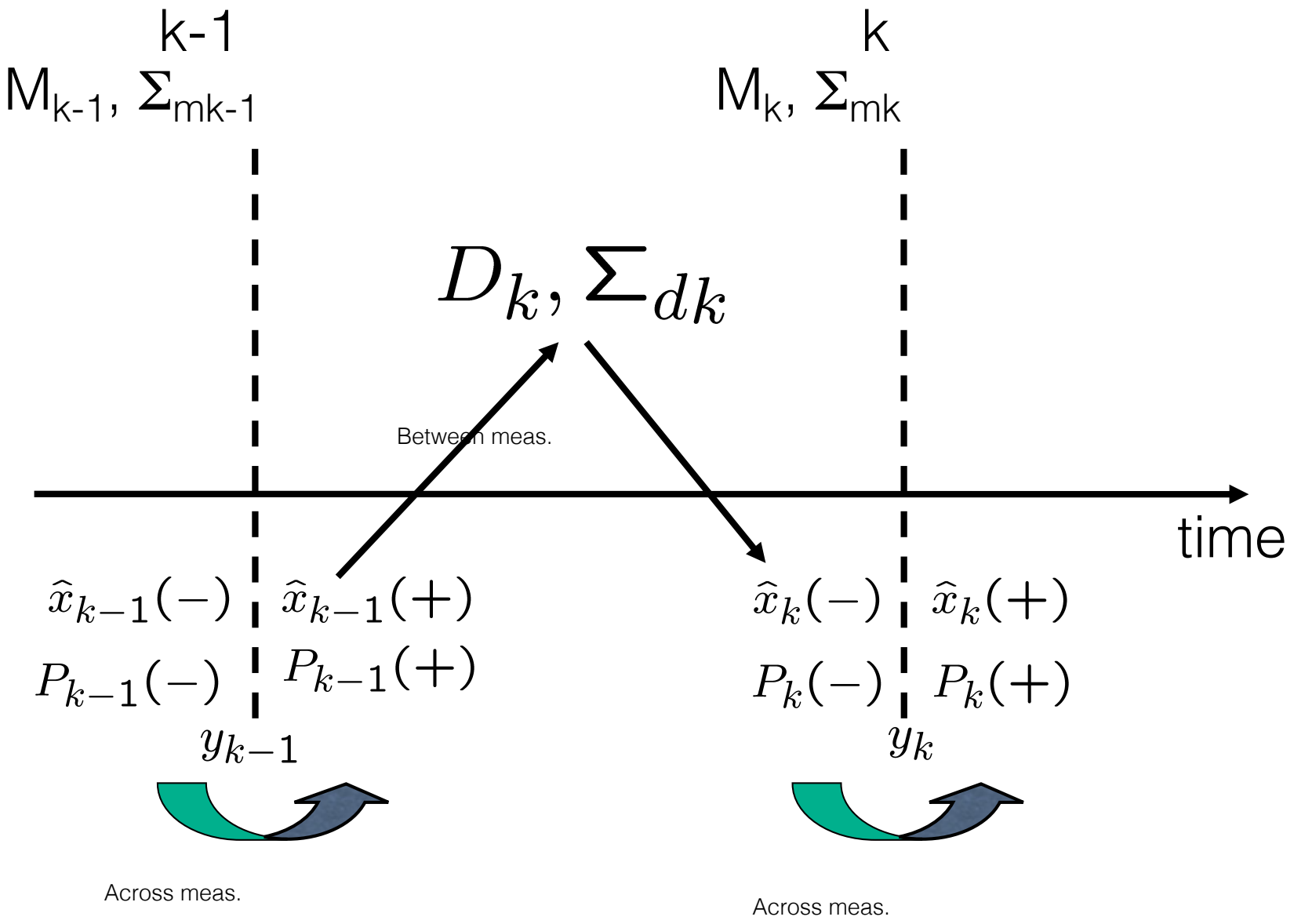
$$\frac{\partial \text{trace} P_k(+)}{\partial K_k} = 2[I - K_k M_k] P_k(-) + 2 K_k \Sigma_{m_k} = 0$$

$$K_k = P_k(-) M_k^T [M_k P_k(-) M_k^T + \Sigma_{m_k}]^{-1}$$

$$P_k(+) = [I - K_k M_k] P_k(-)$$

OK ... that was a lot of algebra! How do we use it?

Timing Diagram



Kalman Filter Algorithm

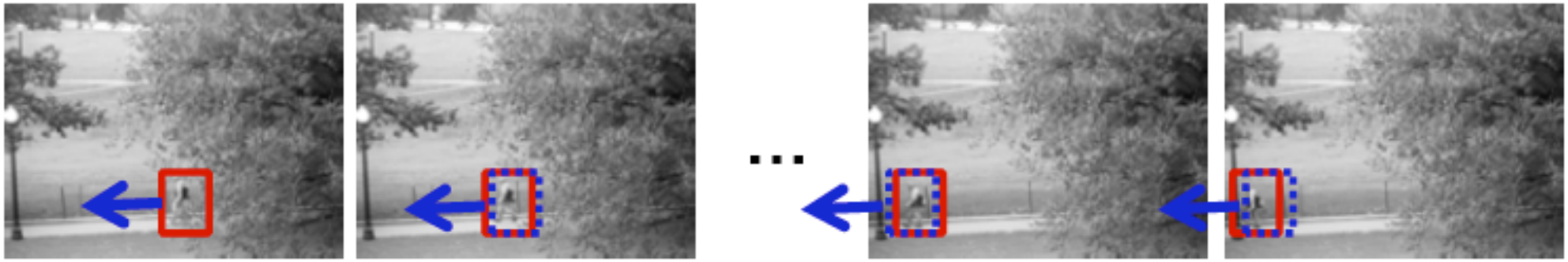
Between Measurements:

$$\begin{aligned} \hat{x}_k(-) &= D_k \hat{x}_{k-1}(+) \\ P_k(-) &= \Sigma_{dk} + D_k P_{k-1}(+) D_k \end{aligned}$$

Across Measurements:

$$\begin{aligned} K_k &= P_k(-) M_k^T [M_k P_k(-) M_k^T + \Sigma_{mk}]^{-1} \\ \hat{x}_k(+) &= \hat{x}_k(-) + K_k [y_k - M_k \hat{x}_k(-)] \\ P_k(+) &= [I - K_k M_k] P_k(-) \end{aligned}$$

Computer Vision Example



Time t Time t+1

Use dynamics to PREDICT and ESTIMATE the position of the target.

Advantages:

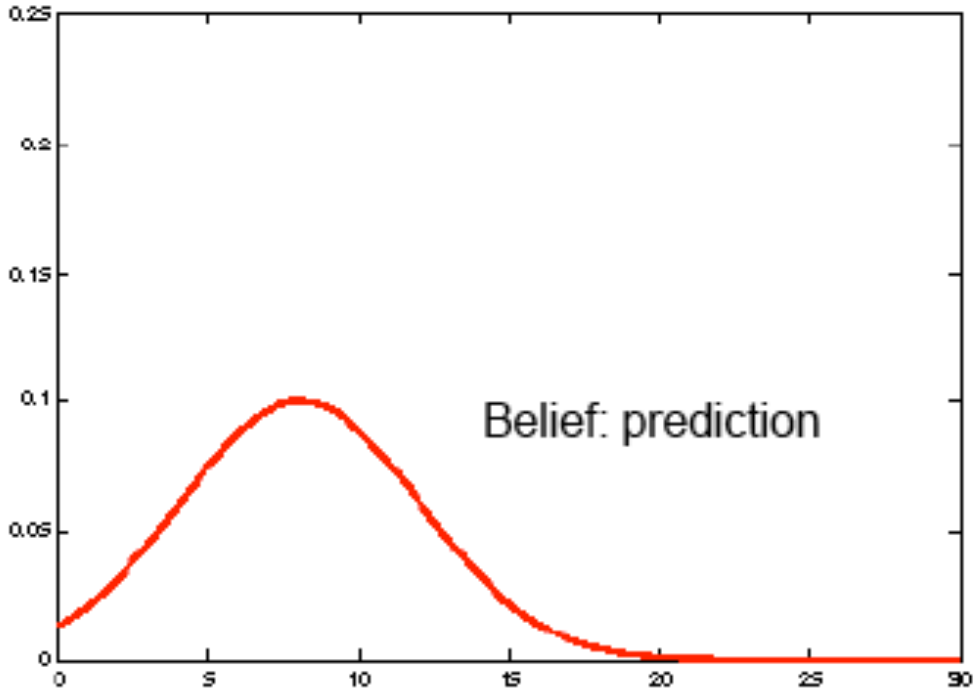
Reduces search space for the target

Improves the estimates of the location since measurements are noisy

Tracking using Prediction

Time t

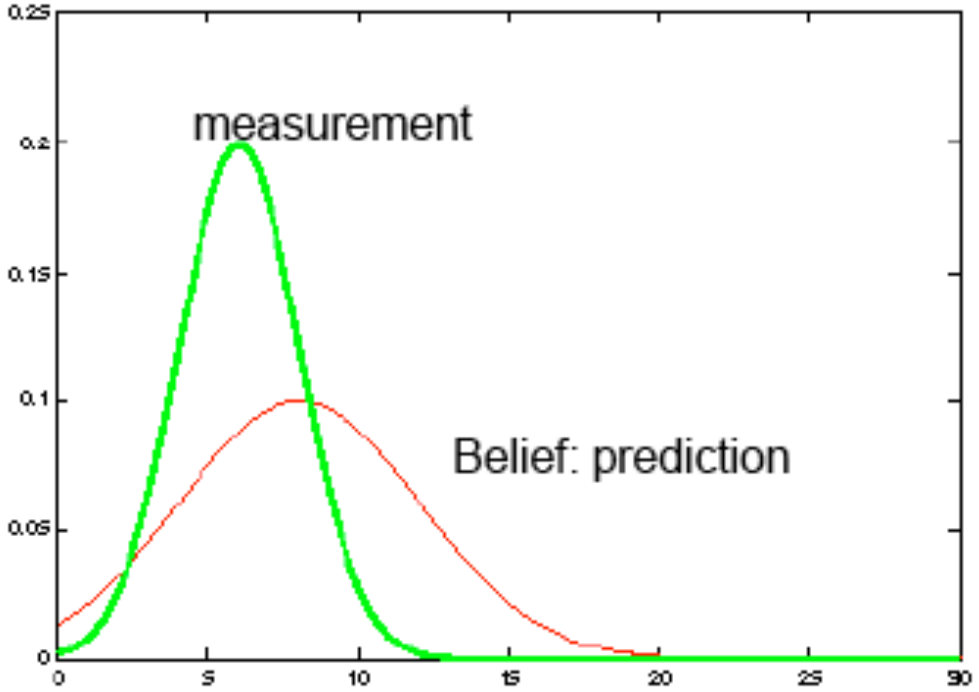
Time t+1



Tracking using Prediction

Time t

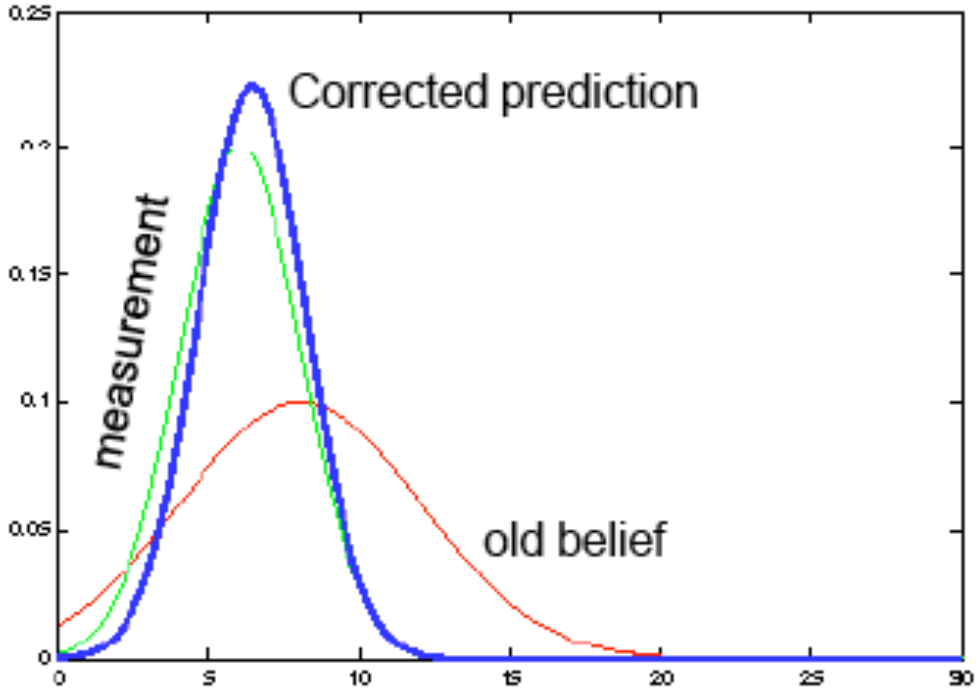
Time t+1



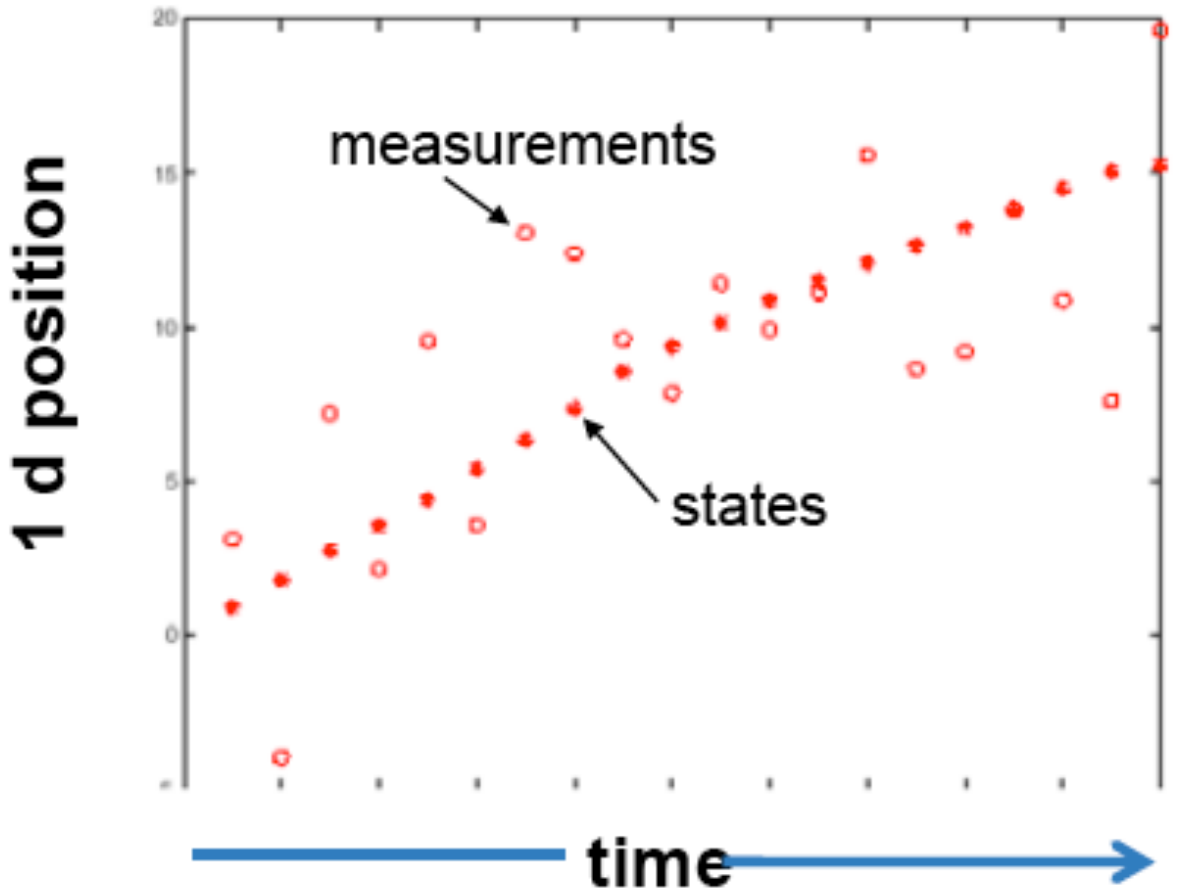
Tracking using Prediction

Time t

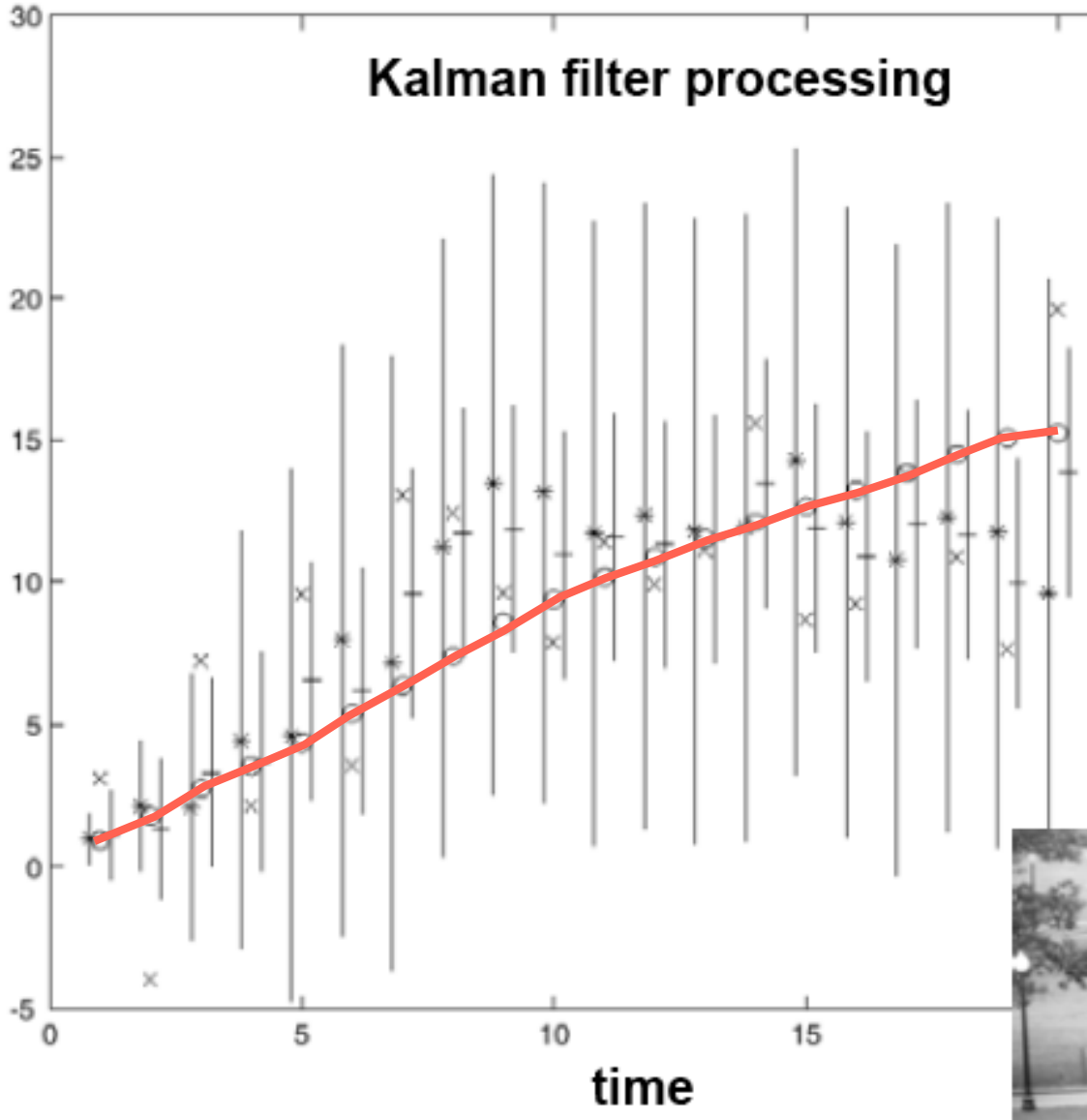
Time t+1



Example: Constant Velocity (1D points)



Constant Velocity Model



o state

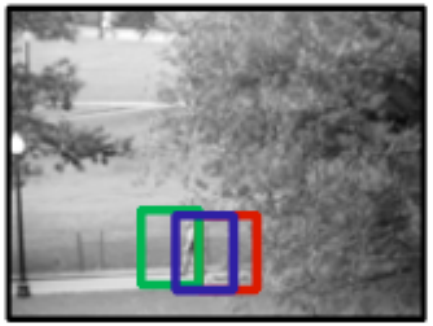
x measurement

* predicted mean estimate

\+ corrected mean estimate

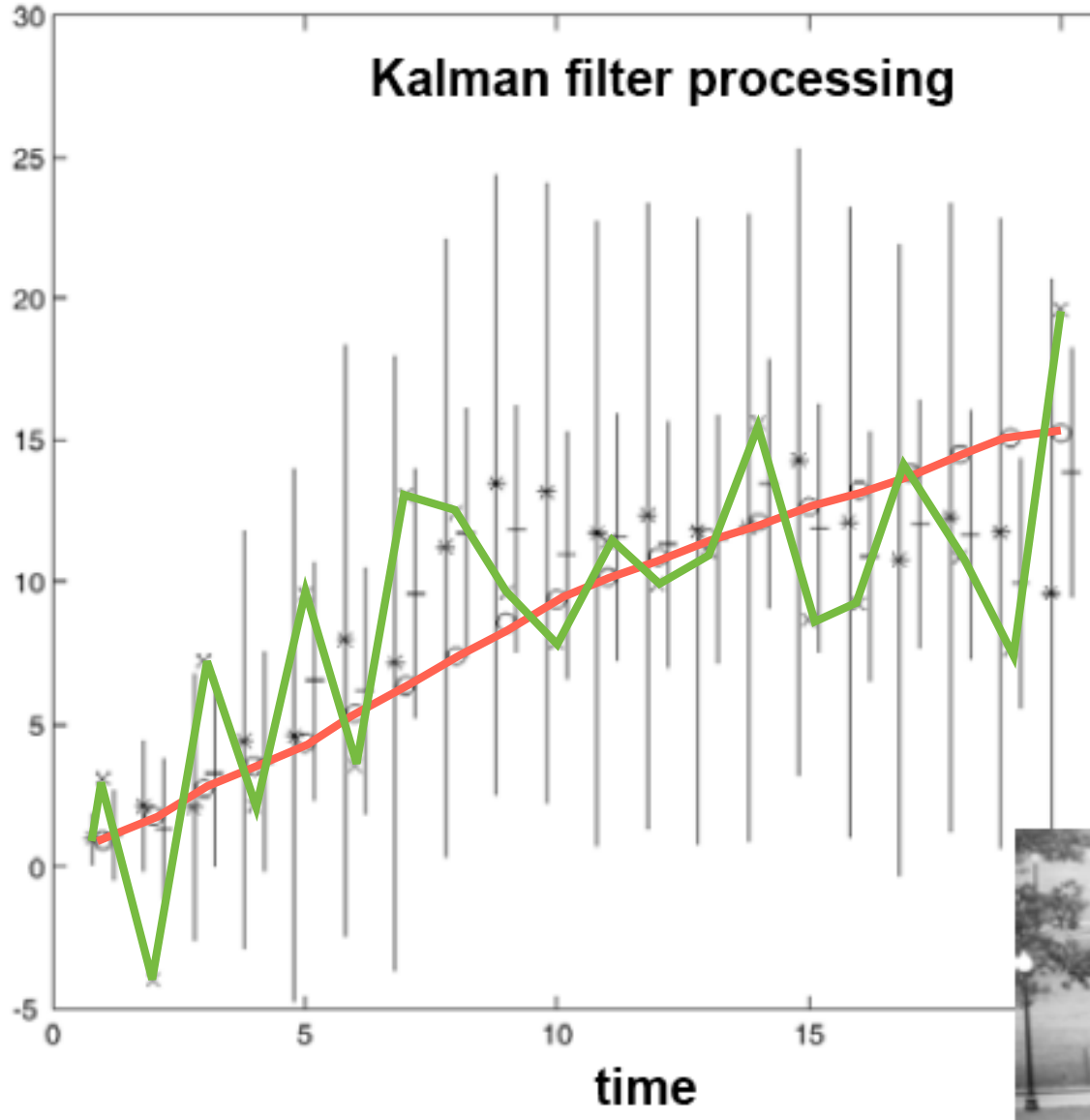
bars: variance estimates before and after measurements

Time t



Time t+1

Constant Velocity Model



o state

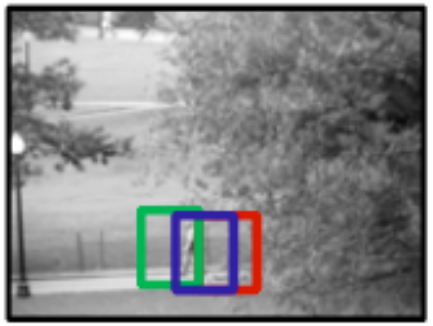
x measurement

* predicted mean estimate

\+ corrected mean estimate

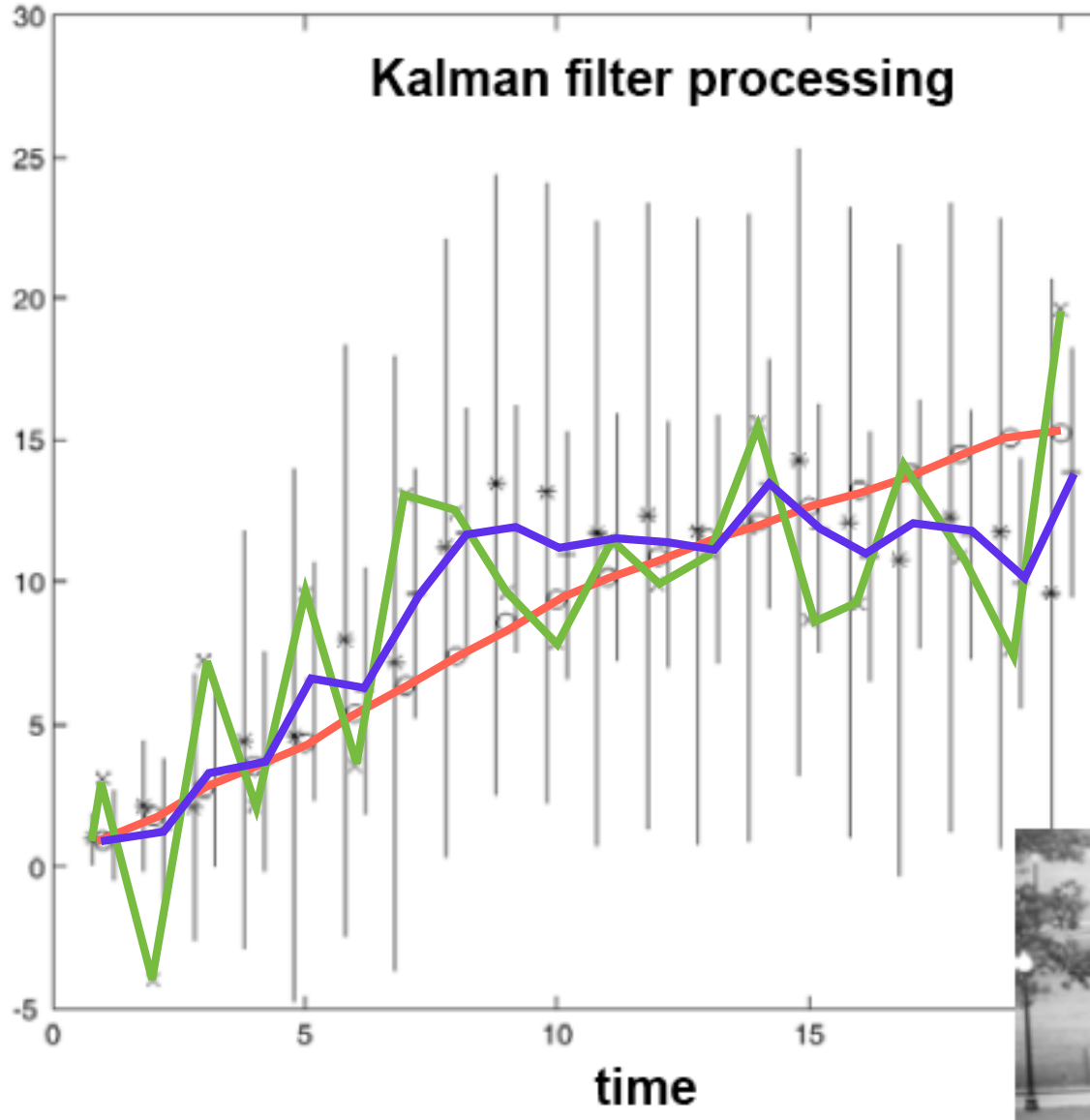
bars: variance estimates before and after measurements

Time t



Time t+1

Constant Velocity Model



o state

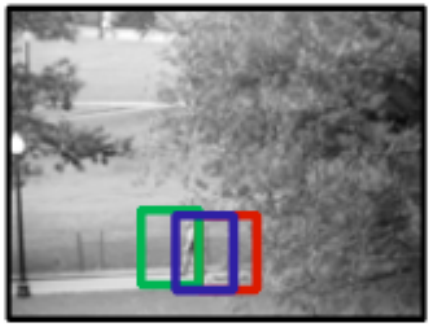
x measurement

* predicted mean estimate

\+ corrected mean estimate

bars: variance estimates before and after measurements

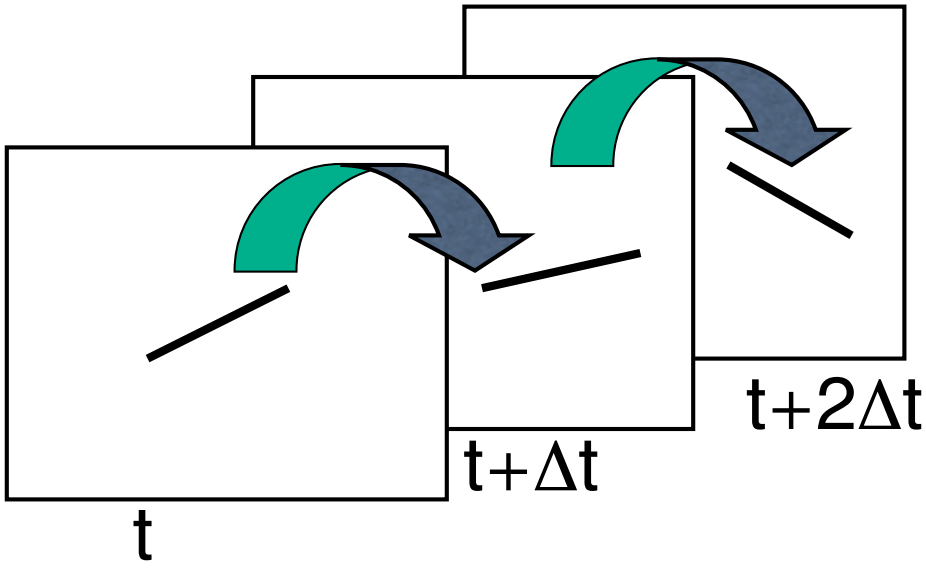
Time t



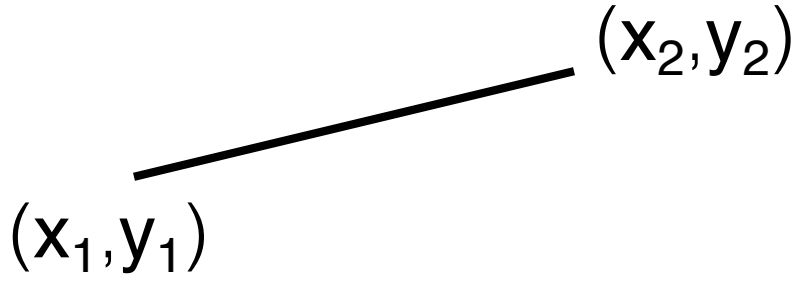
Time t+1

Computer Vision Example

2D Token Tracking: tracking an image segment across frames



A segment can be represented by 4 parameters:



Tracking a Segment

Represent a segment with a vector:

$$r = \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{bmatrix}_{4 \times 1}$$

Define the "state" vector:

$$x = \begin{bmatrix} r \\ \dot{r} \\ \ddot{r} \end{bmatrix}_{12 \times 1}$$

At each time **t**, we measure **r**

$$r_k = \begin{bmatrix} I_4 & 0 & 0 \end{bmatrix} x_k + \nu_k$$

Where do I get the dynamics?

Prior information, modeling, identification

Common assumptions:

- "random" walk
- constant velocity
- constant acceleration

Tracking a Segment

We need the dynamics. Assume for ex. CONSTANT acceleration:

$$x_k = \begin{bmatrix} I_4 & I_4\Delta t & \frac{1}{2}I_4\Delta t^2 \\ 0 & I_4 & I_4\Delta t \\ 0 & 0 & I_4 \end{bmatrix} x_{k-1} + \epsilon_k$$

$$\hat{x}_o = \begin{bmatrix} r_o \\ 0 \\ 0 \end{bmatrix} \quad P_o = \begin{bmatrix} \Sigma_{mo} & 0 & 0 \\ 0 & \lambda^2 I_4 & 0 \\ 0 & 0 & \lambda^2 I_4 \end{bmatrix}$$

Tracking Without Assuming Dynamics

"simple" dynamics might fail if there is prolonged occlusion

We can use tools from system identification (Hankel matrix) to predict future measurements without assuming a dynamic model.

Capturing Dynamics from Experimental Data: The Hankel Matrix

Given a sequence of measurements of d-dimensional vectors:

$$y_1, y_2, y_3 \cdots$$

Its Hankel matrix is defined as:

$$H_y = \begin{bmatrix} y_1 & y_2 & \cdots & y_{n-1} & y_n & \cdots & y_p \\ y_2 & y_3 & \cdots & y_n & y_{n+1} & \vdots & y_{p+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ y_m & y_{m+1} & \cdots & y_{m+n-2} & y_{m+n-1} & \cdots & y_{m+p-1} \end{bmatrix}$$

md x p

Rank(H_y) measures the complexity of the underlying dynamics: after we have "enough" measurements the rank of the Hankel matrix does not increase.

Using a Regressor

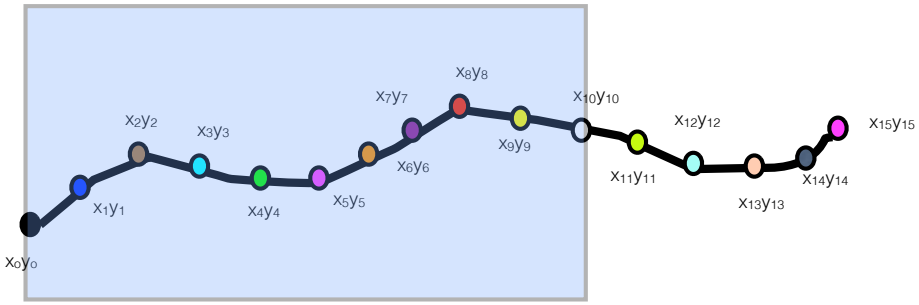
Model the dynamics using a simple regressor:

Regressor:

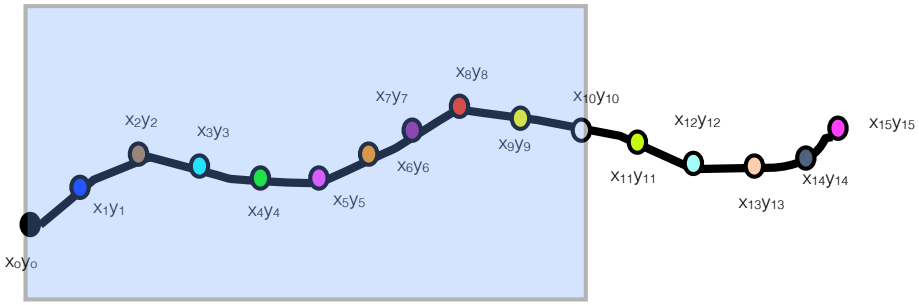
$$y_k = \sum_{i=1}^{\text{Rank}(H_y)} a_i y_{k-i}$$

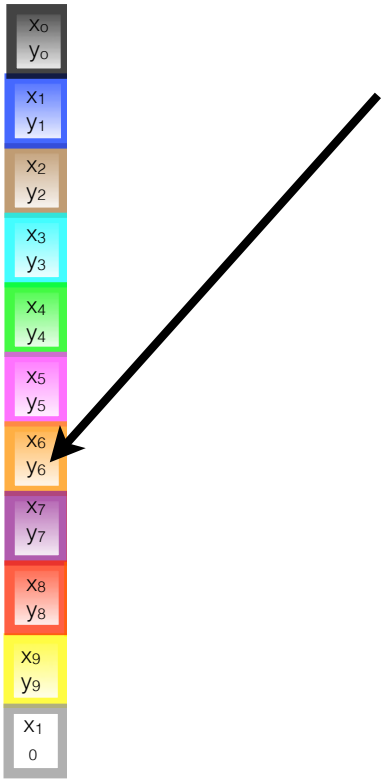
$$H_y = \begin{bmatrix} y_1 & y_2 & \cdots & y_{n-1} & y_n \\ y_2 & y_3 & \cdots & y_n & y_{n+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ y_m & y_{m+1} & \cdots & y_{m+n-2} & y_{m+n-1} \end{bmatrix}$$

Hankel Matrix



Hankel Matrix



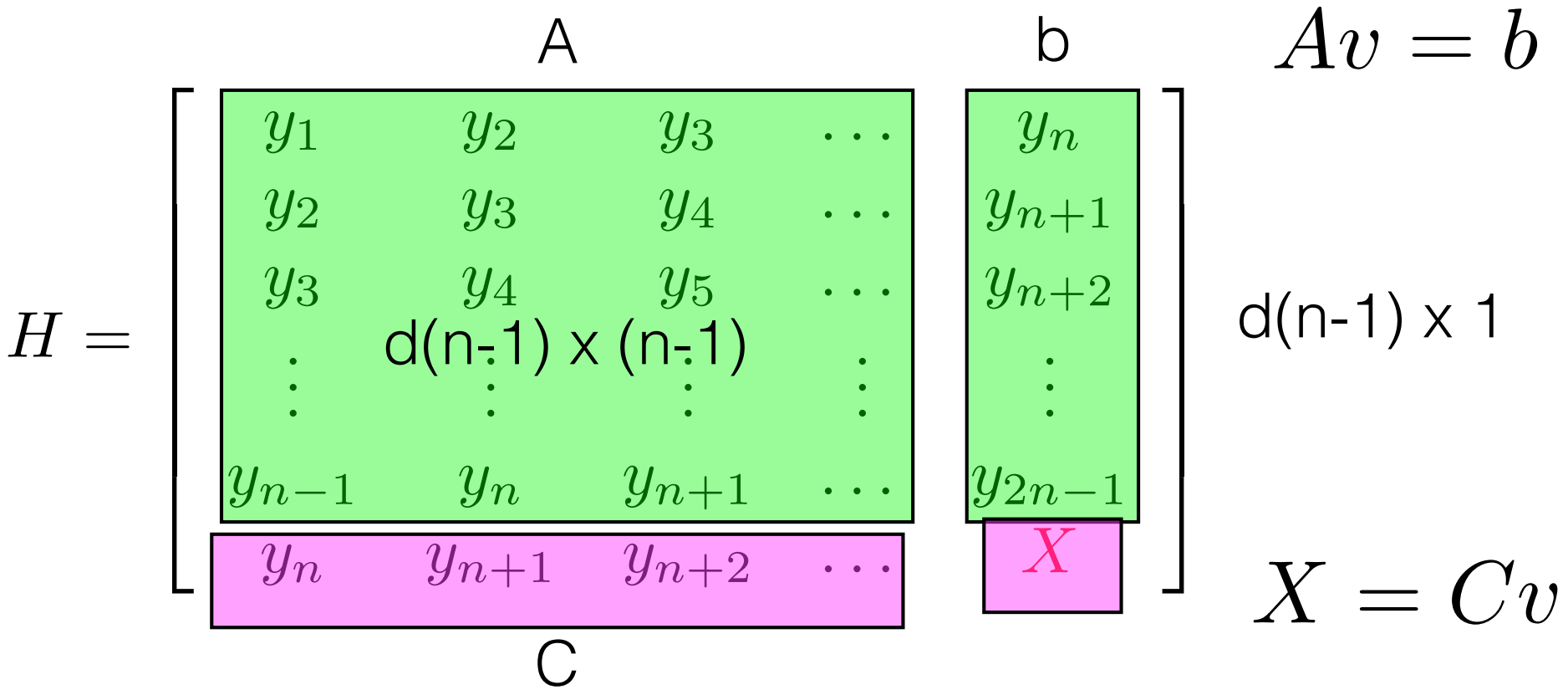


$$\sum_{i=1}^{\text{rank}(H)} a_i \begin{bmatrix} x_{k-i} \\ y_{k-i} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \end{bmatrix}$$

$$\text{Rank}(H) = n = \text{Complexity of Dynamics}$$

Predicting the next measurement

The new measurement should not increase the complexity of the dynamics:



The last column must be a linear combination of the previous ones. **(But we should know the order of the system n-1).**

Data Prediction: Receding Horizon Tracking

Particle Kalman RHRM

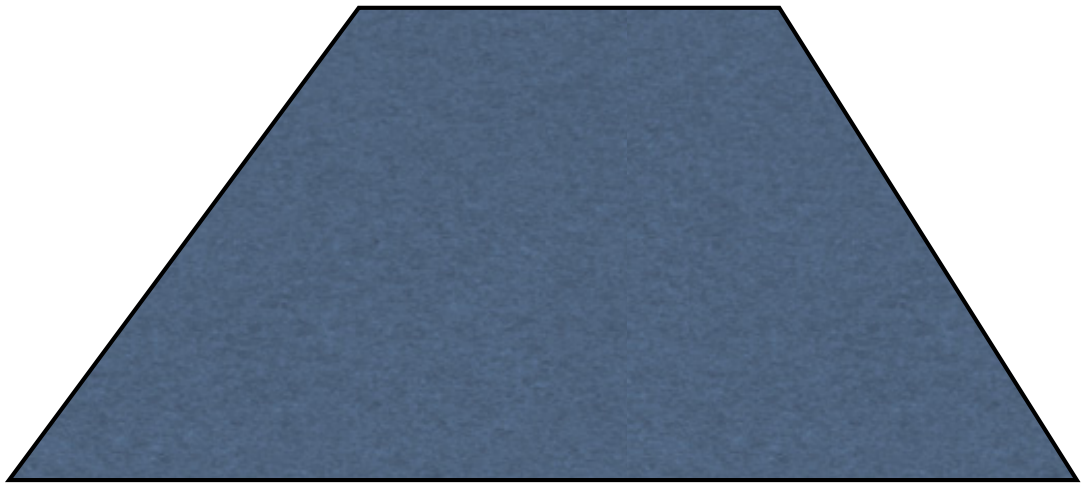
Hankel approach:

- Assemble Hankel matrix with noise

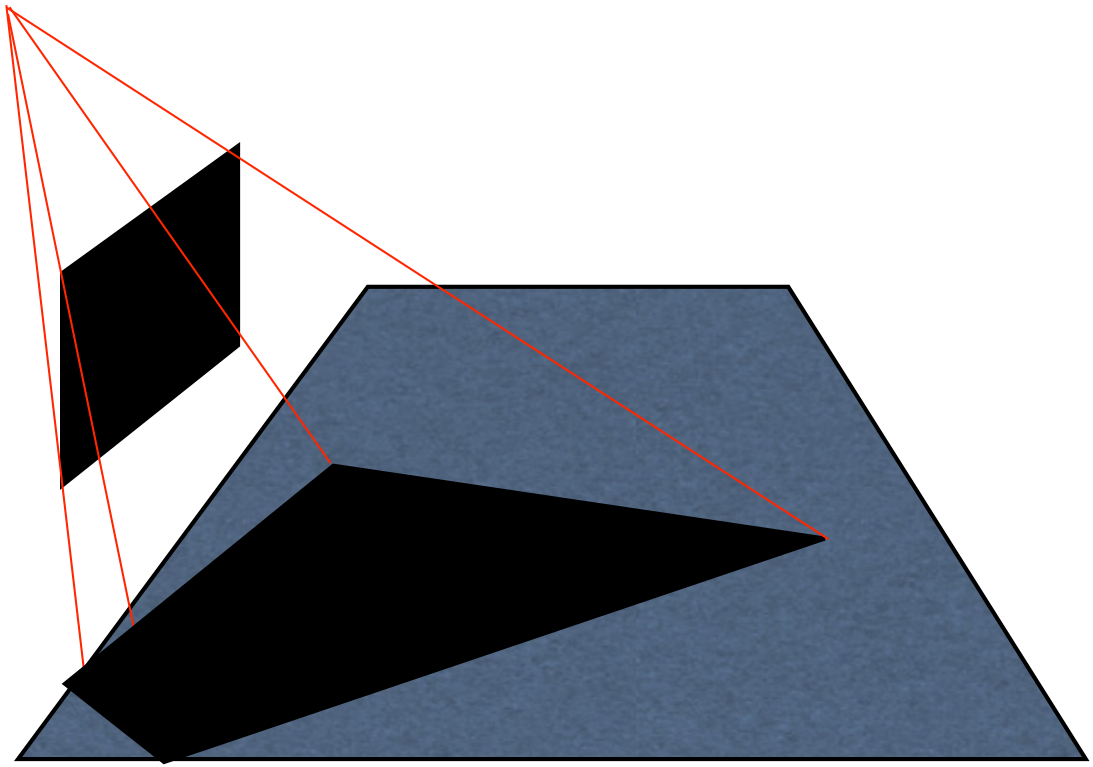
$$H = \begin{bmatrix} y_1+\eta_1 & y_2+\eta_2 & y_3+\eta_3 & \cdots & y_n+\eta_n \\ y_2+\eta_2 & y_3+\eta_3 & y_4+\eta_4 & \cdots & y_{n+1}+\eta_{n+1} \\ y_3+\eta_3 & y_4+\eta_4 & y_5+\eta_5 & \cdots & y_{n+2}+\eta_{n+2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ y_n+\eta_n & y_{n+1}+\eta_{n+1} & y_{n+2}+\eta_{n+2} & \cdots & y_{2n-1}+\eta_{2n-1} \end{bmatrix}$$

- Minimize rank(H) wrt noise
- Predict next measurement/update

Surveillance of Large Public Spaces

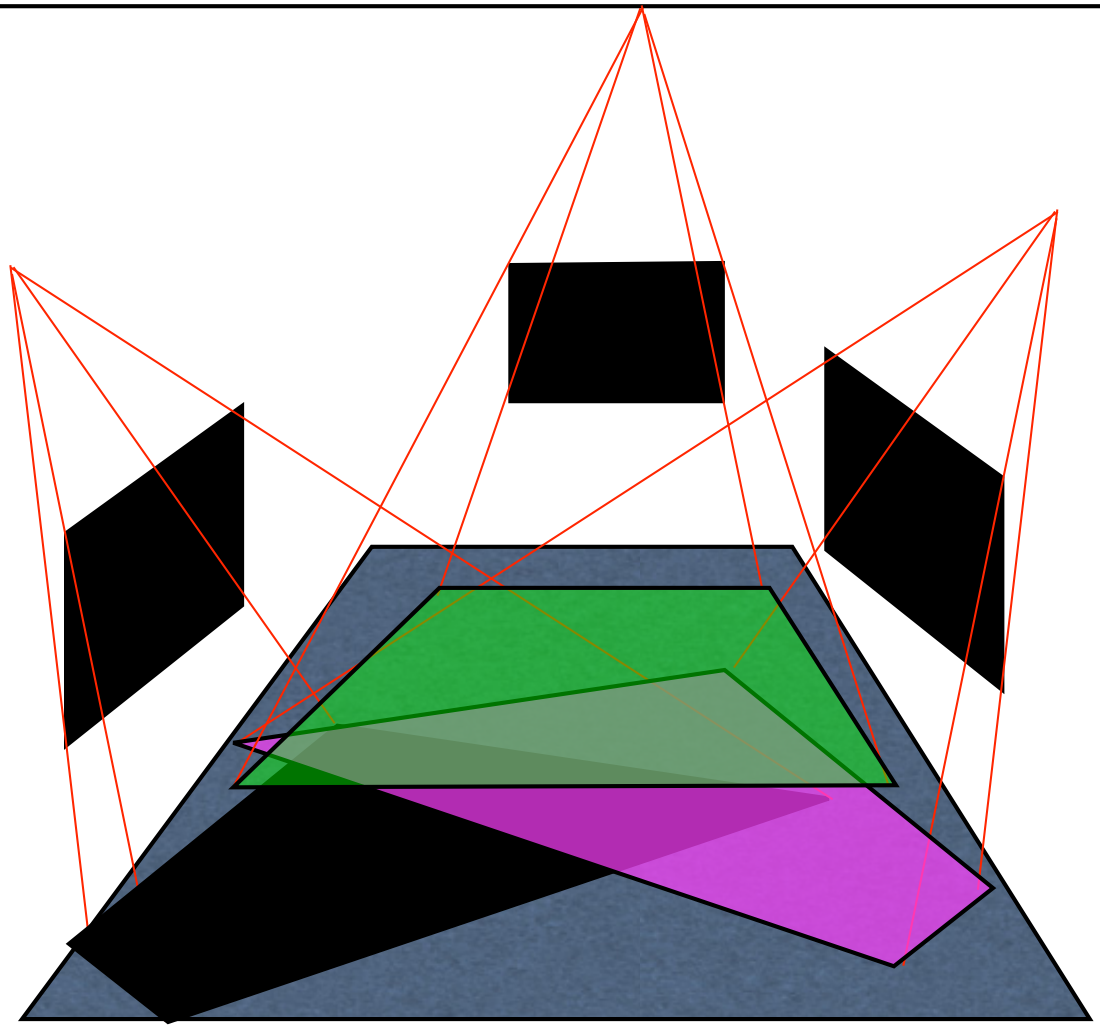


Surveillance of Large Public Spaces



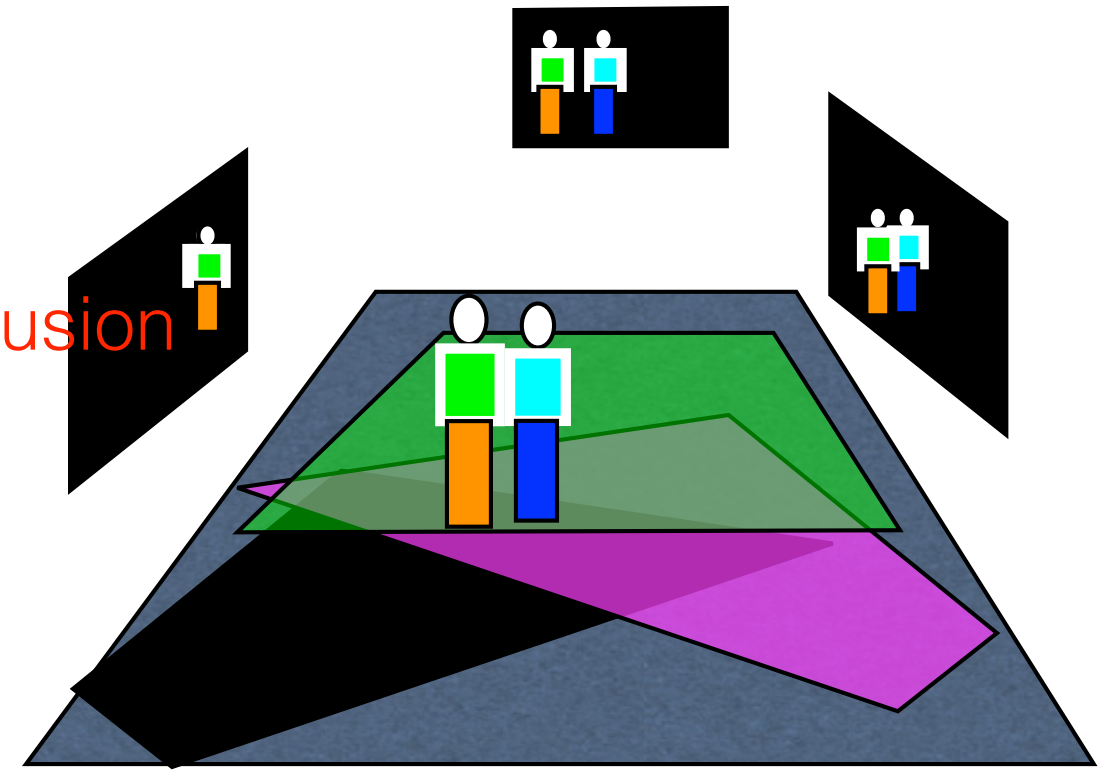
Cameras have a limited field of view

Surveillance of Large Public Spaces



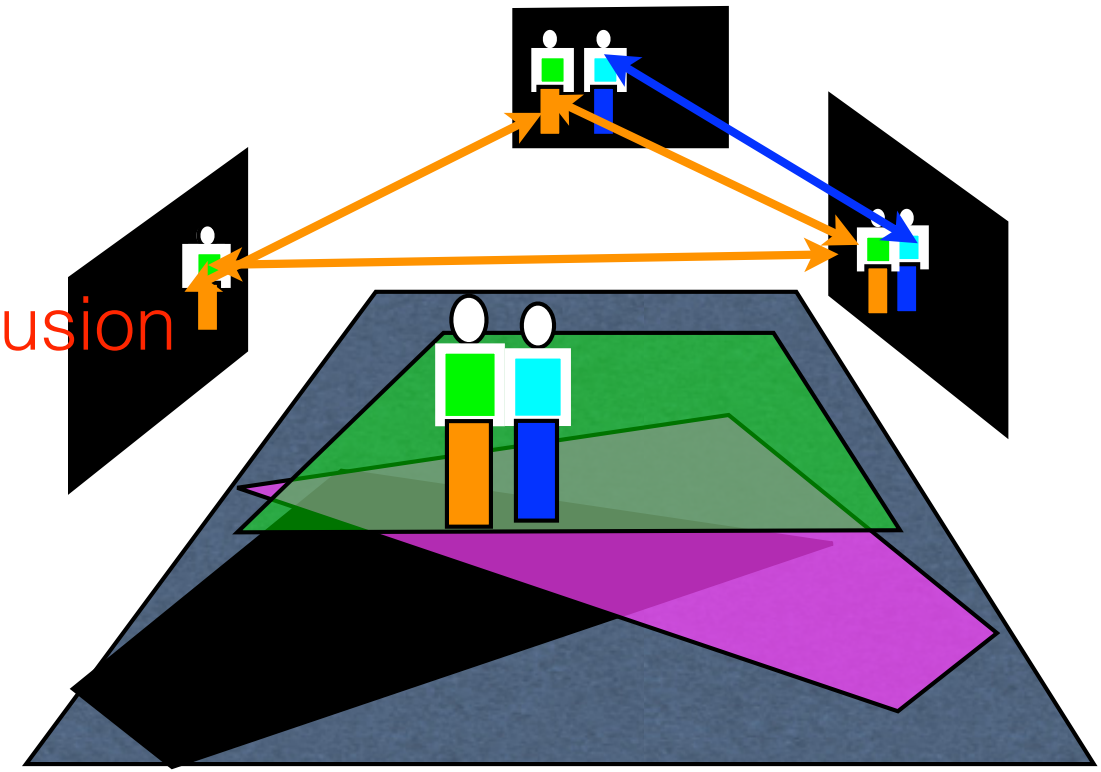
Use multiple cameras to cover a large area.

Surveillance of Large Public Spaces



If a target is occluded in one camera, it is still visible in others.

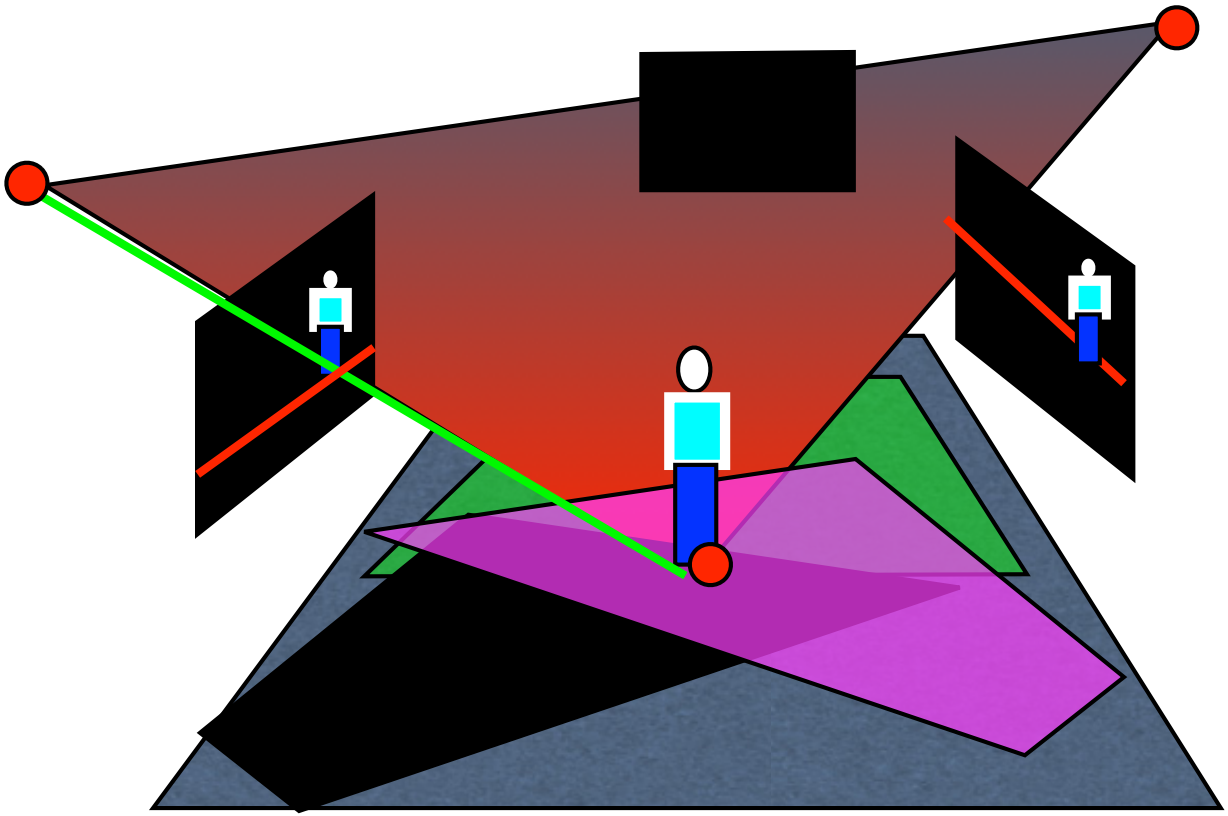
Surveillance of Large Public Spaces



We must solve the correspondence problem.

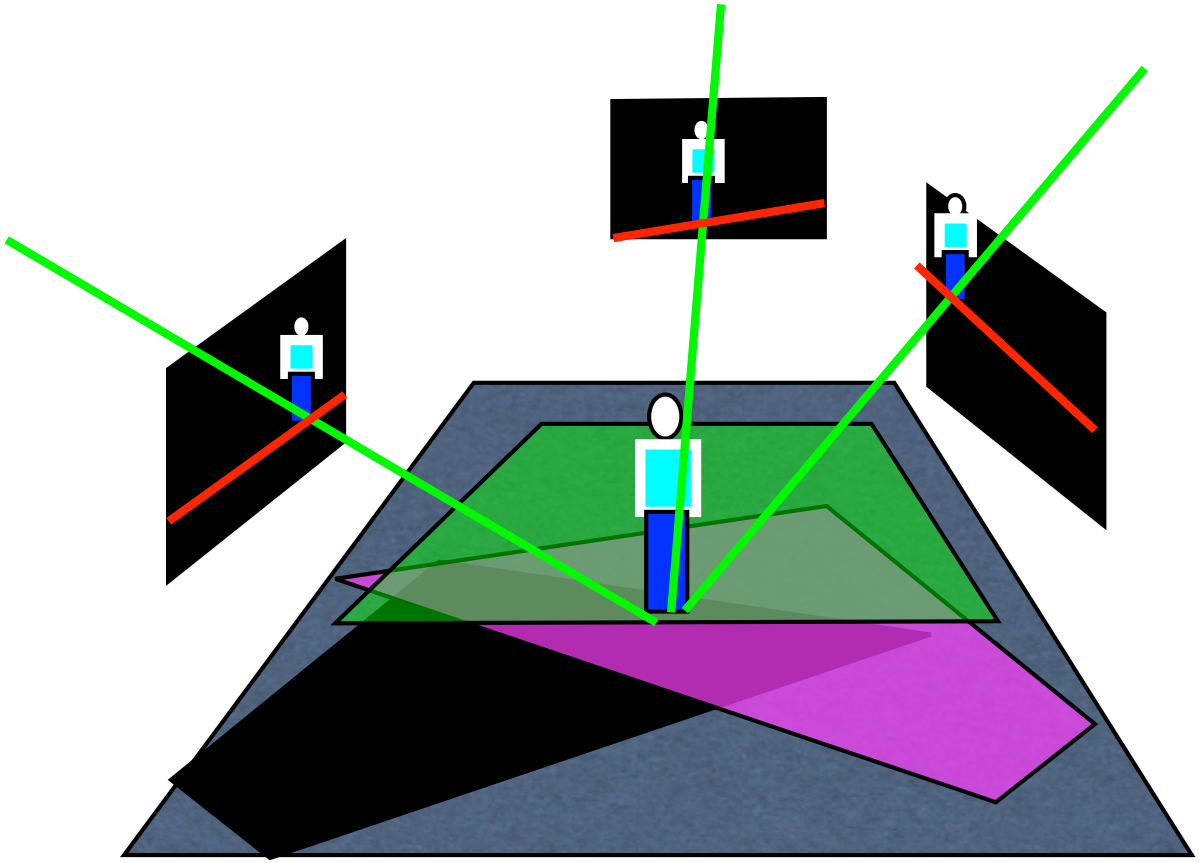
Surveillance of Large Public Spaces

Geometric Constraints: Epipolar Geometry



Surveillance of Large Public Spaces

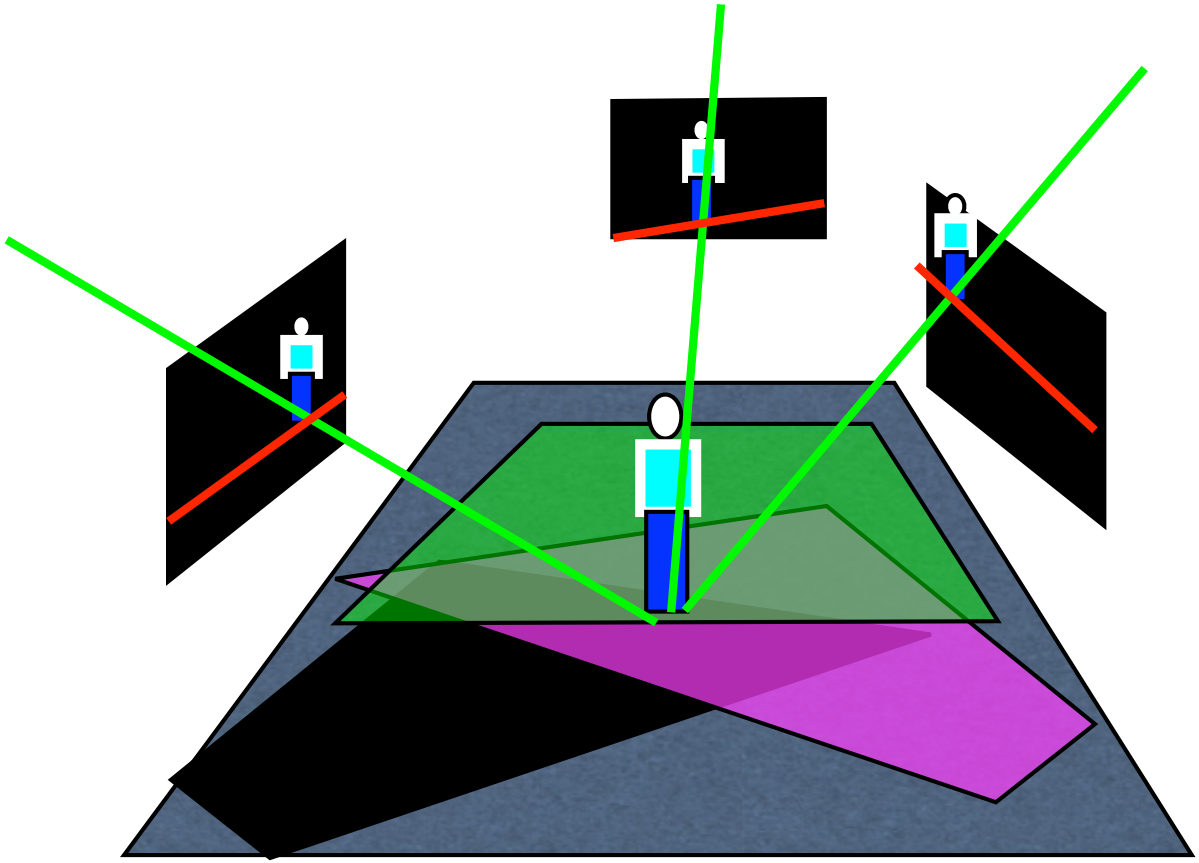
Geometric Constraints: Epipolar Geometry



Corresponding features must lie in corresponding epipolar lines.

Surveillance of Large Public Spaces

3D Assisted Tracker Using Epipolar Geometry:



Corresponding features must lie in corresponding epipolar lines.

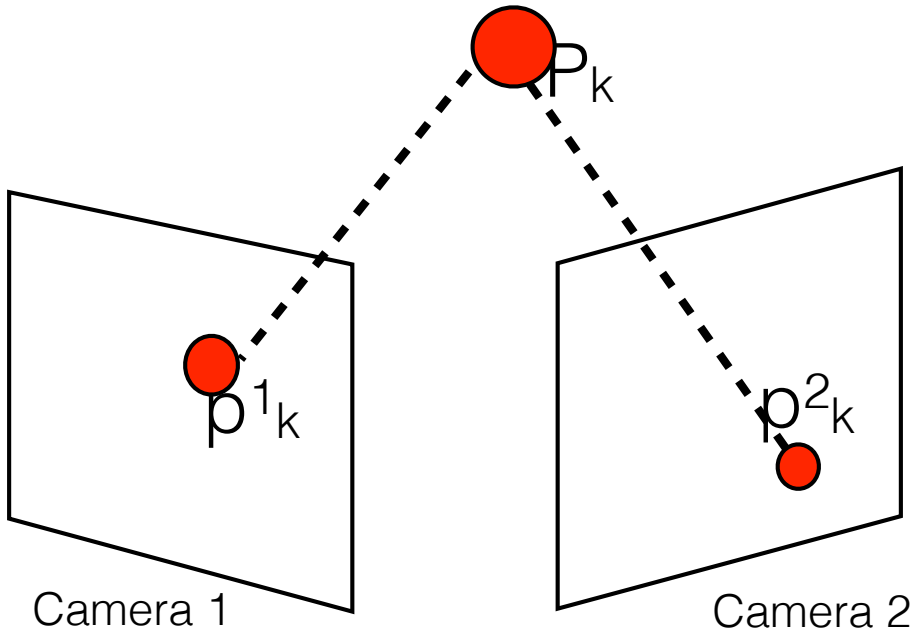
Surveillance of Large Public Spaces

Geometric Constraints: Epipolar Geometry

Tracking failures due to long occlusions

Dynamic-based View Invariant for Multi-camera Tracking

$$p_k^{(1)} = \Pi_1 P_k$$



$$p_k^{(2)} = \Pi_2 P_k$$

Affine projection model:

$$\Pi_j \quad 2 \times 4 \text{ matrix}$$

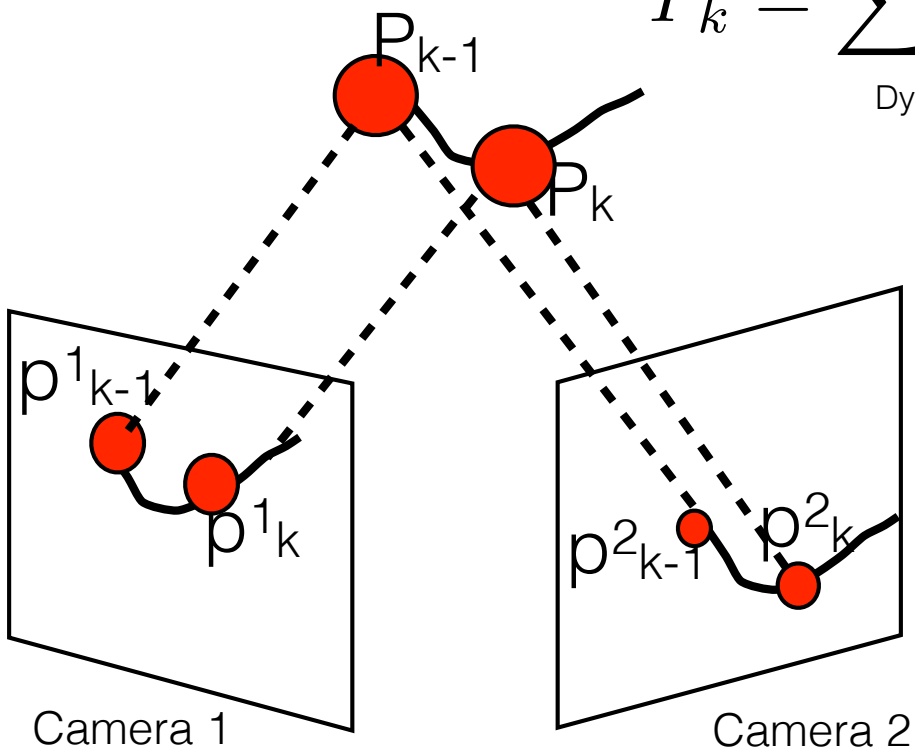
Dynamic-based View Invariant for Multicamera Tracking

$$P_k = \sum a_i P_{k-i}$$

Dynamics Regressor in 3D

$$p_k^{(1)} = \Pi_1 P_k$$

$$p_k^{(2)} = \Pi_2 P_k$$



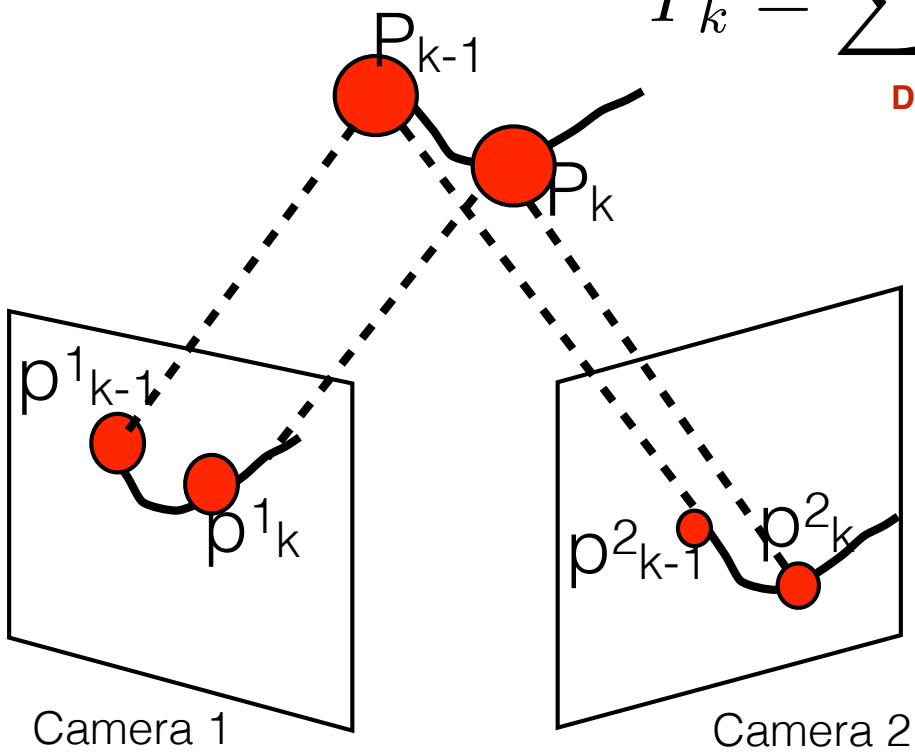
Dynamic-based View Invariant for Multicamera Tracking

$$P_k = \sum a_i P_{k-i}$$

Dynamics Regressor in 3D

$$p_k^{(1)} = \Pi_1 P_k$$

$$p_k^{(2)} = \Pi_2 P_k$$



$$p_k^{(j)} = \Pi_j P_k = \Pi_j \sum a_i P_{k-i} = \sum a_i \Pi_j P_{k-i} = \sum a_i p_{k-i}^{(j)}$$

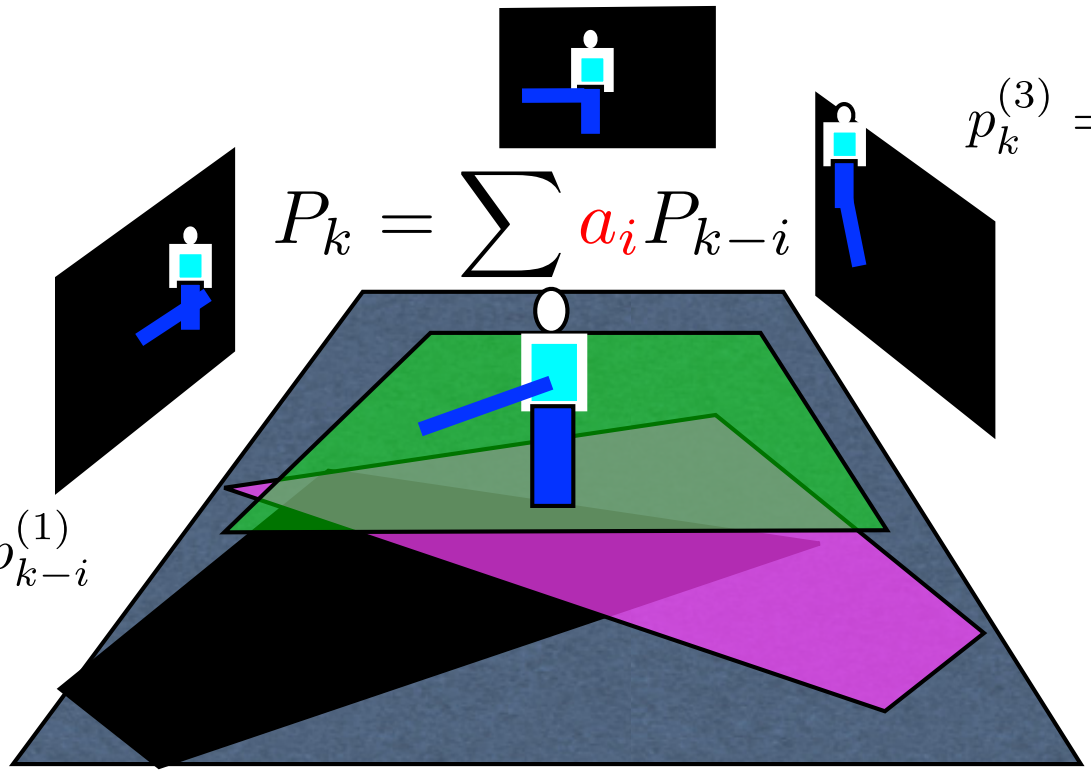
Dynamic Constraints: Trajectories Subspace

$$p_k^{(2)} = \sum a_i p_{k-i}^{(2)}$$

$$p_k^{(3)} = \sum a_i p_{k-i}^{(3)}$$

$$P_k = \sum a_i P_{k-i}$$

$$p_k^{(1)} = \sum a_i p_{k-i}^{(1)}$$



All corresponding trajectories live in the subspace orthogonal to their regressor.

Dynamic-based View Invariant for Multicamera Tracking

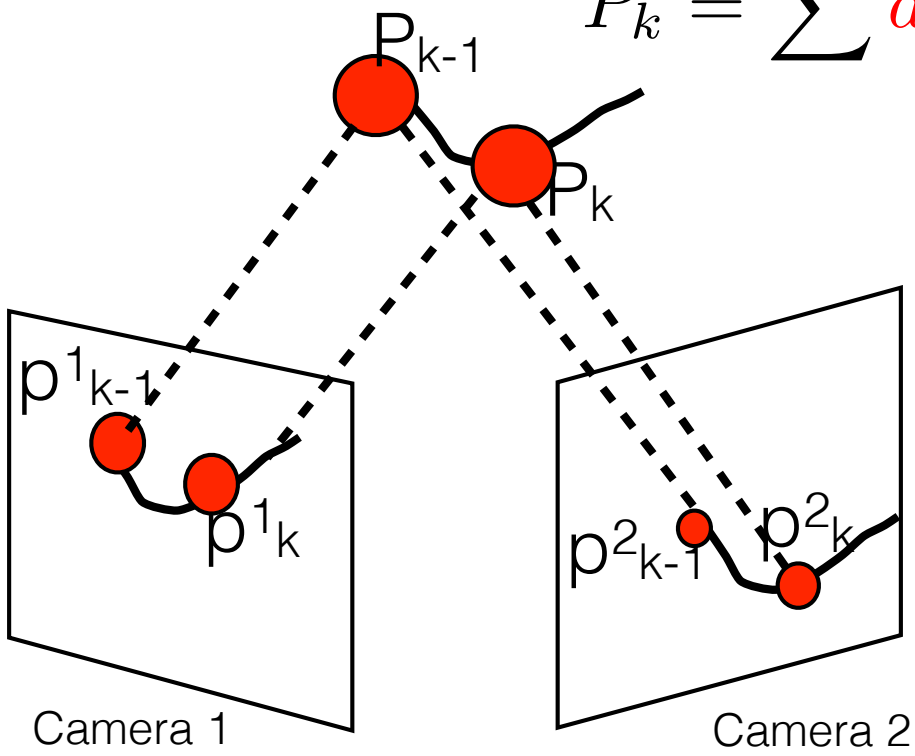
$$P_k = \sum a_i P_{k-i}$$

$$p_k^{(1)} = \Pi_1 P_k$$

$$p_k^{(1)} = \sum a_i p_{k-i}^{(1)}$$

$$p_k^{(2)} = \Pi_2 P_k$$

$$p_k^{(2)} = \sum a_i p_{k-i}^{(2)}$$

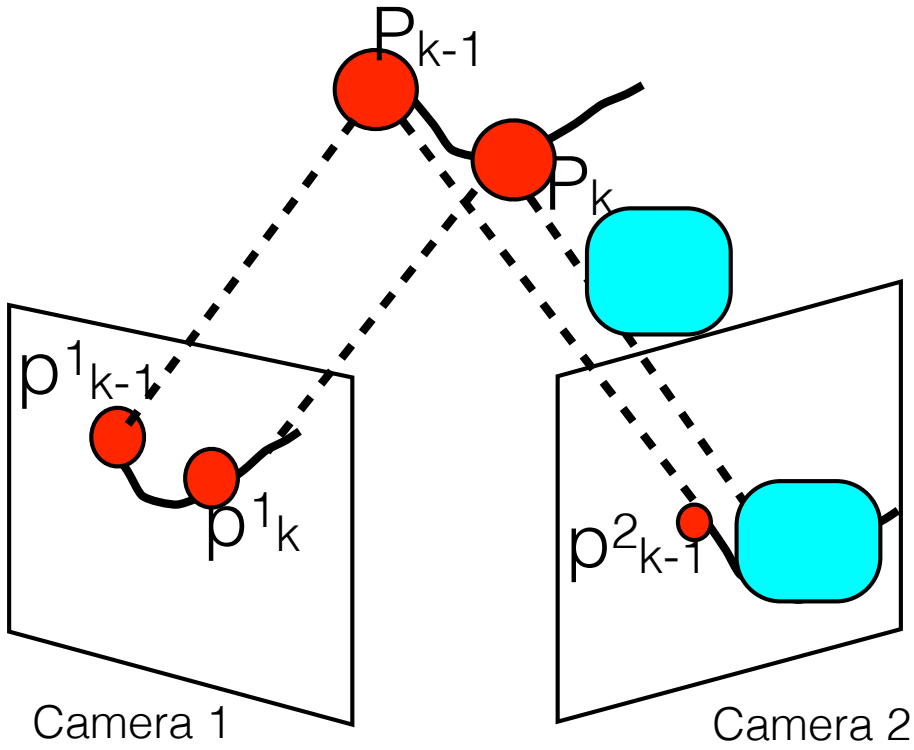


$$\begin{bmatrix} p_k^{(1)} \\ p_k^{(2)} \end{bmatrix} = \begin{bmatrix} p_{k-1}^{(1)} & p_{k-2}^{(1)} & \cdots \\ p_{k-1}^{(2)} & p_{k-2}^{(2)} & \cdots \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix}$$

Trajectories' subspaces can be jointly estimated

Dynamic-based View Invariant for Multicamera Tracking

$$p_k^{(1)} = \sum a_i p_{k-i}^{(1)}$$



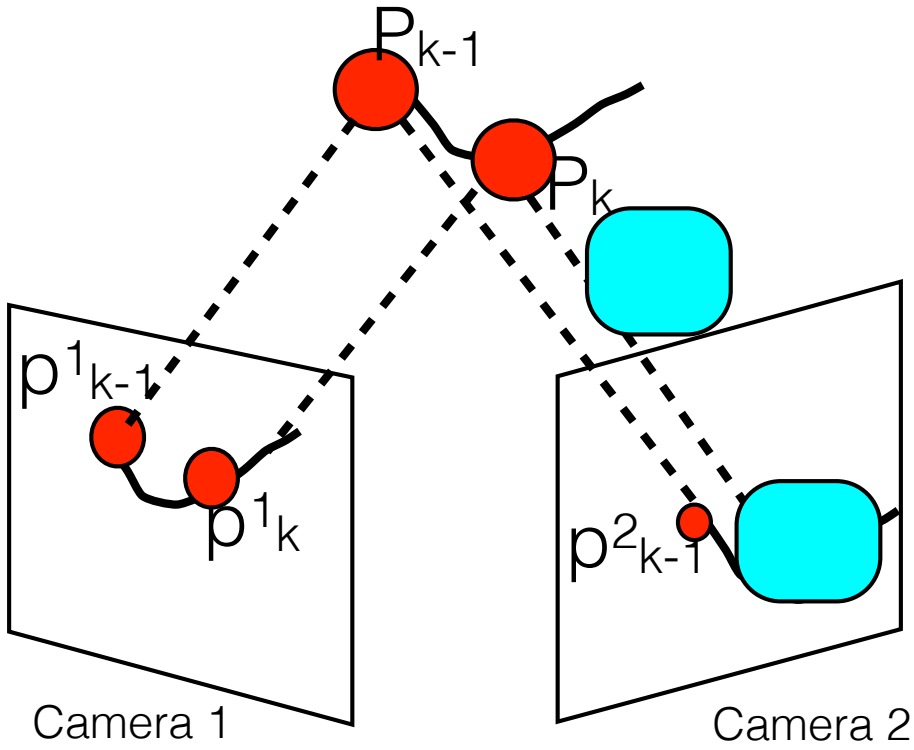
$$p_k^{(2)} = \sum a_i p_{k-i}^{(2)}$$

$$\begin{bmatrix} p_k^{(1)} \\ p_k^{(2)} \end{bmatrix} = \begin{bmatrix} p_{k-1}^{(1)} & p_{k-2}^{(1)} & \cdots \\ p_{k-1}^{(2)} & p_{k-2}^{(2)} & \cdots \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix}$$

If Camera 2 is occluded, Camera 1 takes over

Dynamic-based View Invariant for Multicamera Tracking

$$p_k^{(1)} = \sum a_i p_{k-i}^{(1)}$$

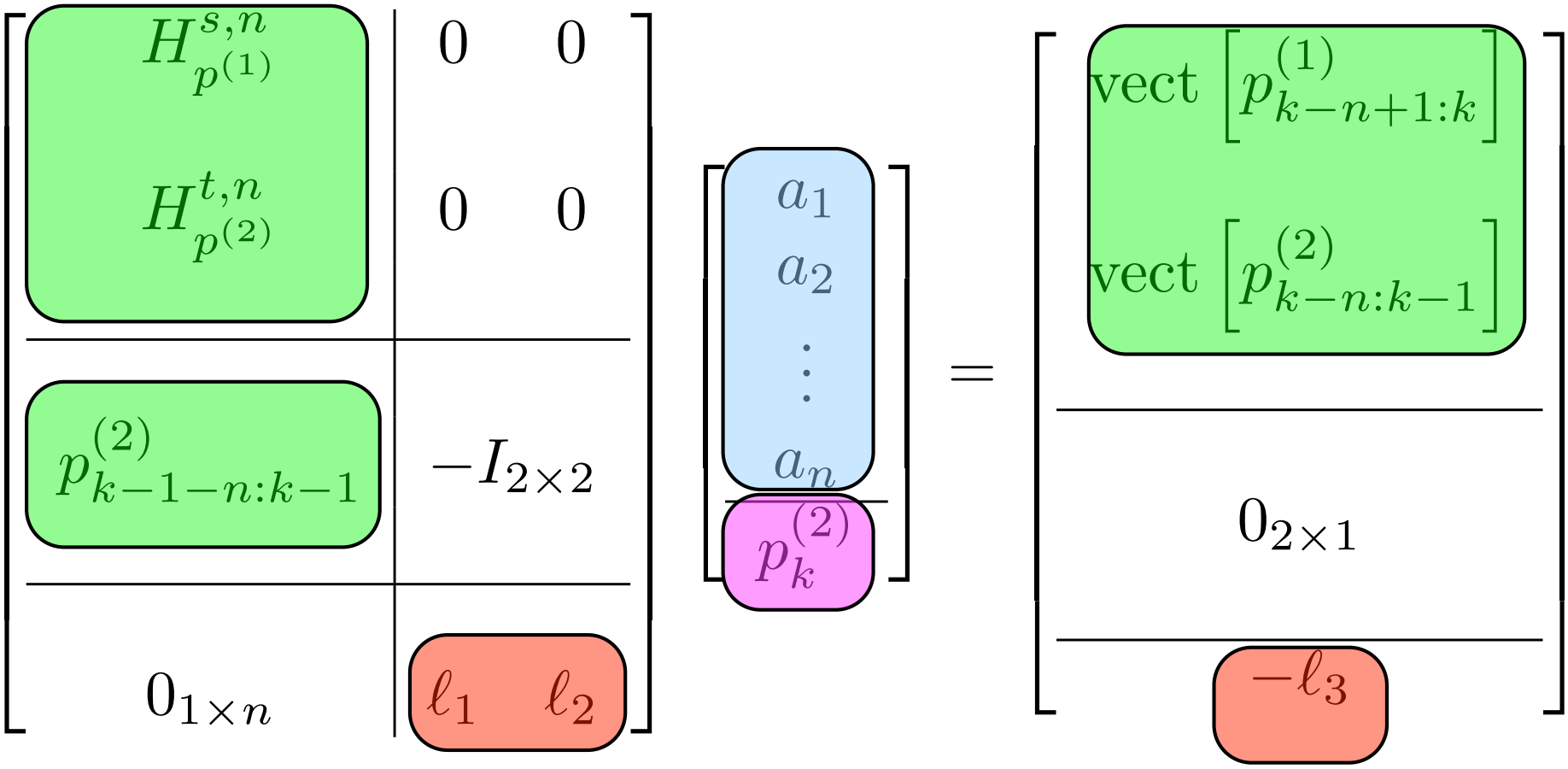


$$p_k^{(2)} = \sum a_i p_{k-i}^{(2)}$$

$$\begin{bmatrix} p_k^{(1)} \\ (2) \end{bmatrix} = \begin{bmatrix} p_{k-1}^{(1)} & p_{k-2}^{(1)} & \cdots \\ (2) & (2) & \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix}$$

If Camera 2 is occluded, Camera 1 takes over

Jointly use of geometry and dynamics



where

$$\begin{bmatrix} p_k^{(1)^T} & 1 \end{bmatrix} F = \begin{bmatrix} l_1 & l_2 & l_3 \end{bmatrix}$$

Surveillance of Large Public Spaces

Dynamic Constraints: Trajectories Subspace

If a target is occluded in one camera, it is still be visible in others.

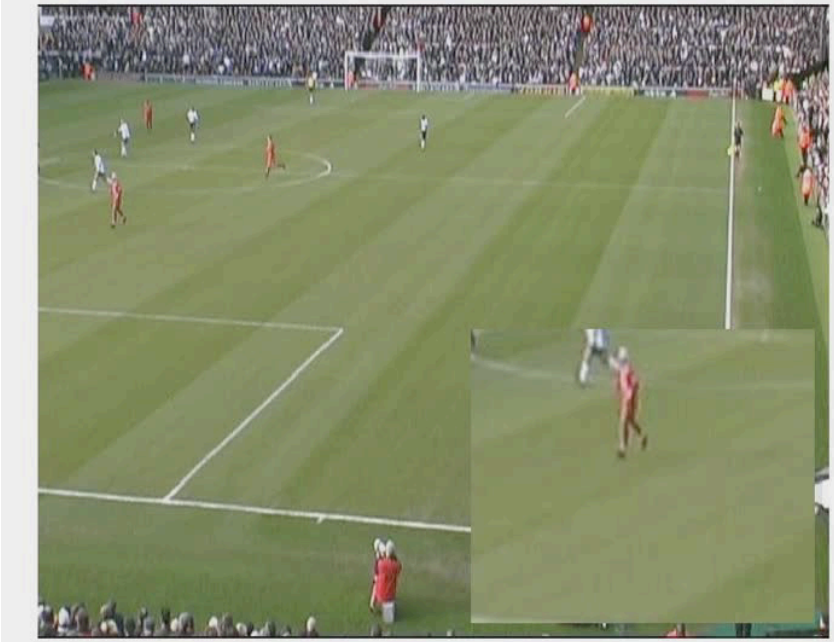
Surveillance of Large Public Spaces

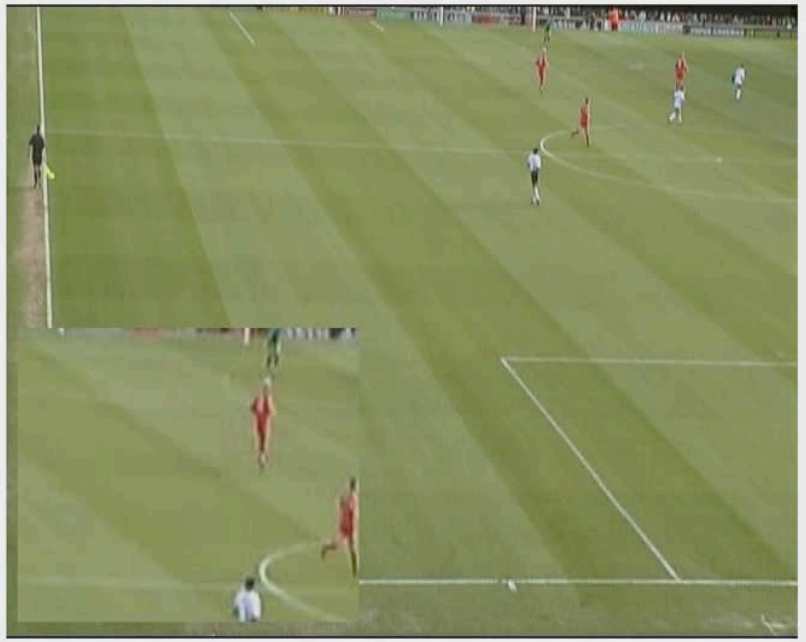
Dynamic Constraints: Trajectories Subspace

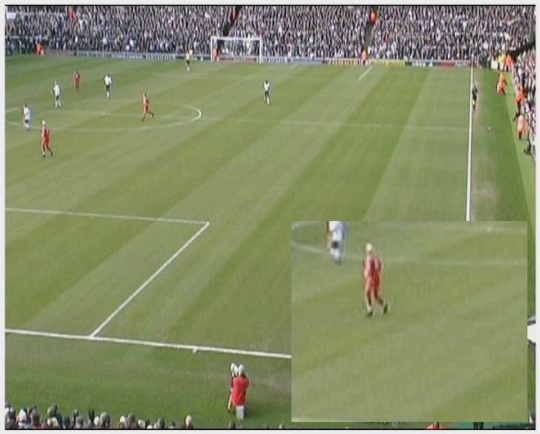
If a target is occluded in one camera, it is still be visible in others.

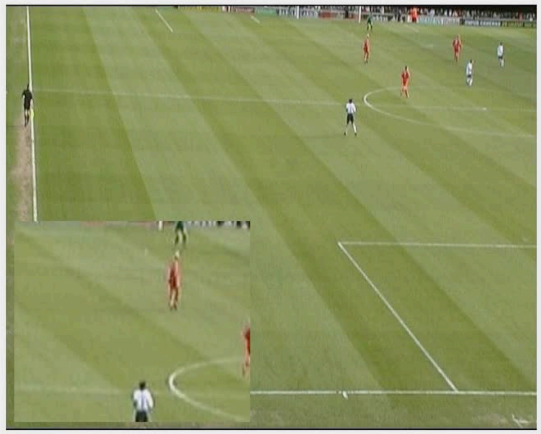
Multi-camera Tracking

Our Method









Without using dynamic constraints